VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Bohaté internetové aplikace a technologie. NET

Rich Internet Applications and .NET Technologies

2011 Marek Jan

Poděkování:

Rád bych poděkoval Ing. Michalu Radeckému za odbornou pomoc a cenné rady při tvorbě diplomové práce. Dále bych chtěl poděkovat své rodině a přátelům za podporu při studiu.

Prohlášení:

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 5. května 2011 ……

Abstrakt

Práce se zabývá zmapováním a ověřením možností tvorby bohatých internetových aplikací v prostředí internetu. Jsou zde zkoumány a porovnávány nejznámější technologie pro tvorbu bohatých internetových aplikací, a to AJAX, Adobe Flex, Microsoft Silverlight a JavaFX. Následně je popsán návrh a implementace aplikace, která obsahuje různé testovací případy vytvořené pomocí rozličných grafických rozhraní a prvků. Jejím cílem je automaticky otestovat, zda jsou dané testovací případy pro uživatele snadno ovladatelné a nečiní jim potíže. Výstupy dané testovací aplikace jsou poté zanalyzovány a statisticky zpracovány. Závěr je zhodnocení a návrh nejlepších postupů pro tvorbu bohatých internetových aplikací s ohledem na výsledky analýz a budoucích trendů.

Klíčová slova: Bohaté internetové aplikace, AJAX, Adobe Flex, Microsoft Silverlight, JavaFX, uživatelské rozhraní, Ribbon

Abstract

The work deals with the mapping and verification the possibility of creating rich Internet applications on the Internet. There are examined and compared the best-known technology for creating rich Internet applications, AJAX, Adobe Flex, Microsoft Silverlight and JavaFX. Then describe the design and implementation of an application that contains different test cases, created by various graphical interfaces and features. Its goal is to automatically test if the test cases are user-friendly and makes no problems. The outputs of the test applications are then processed and statistically analyzed. The conclusion of the evaluation and design best practices for building rich internet applications with respect to the analysis results and future trends.

Keywords: Rich Internet Applications, AJAX, Adobe Flex, Microsoft Silverlight, JavaFX, graphical user interface, Ribbon

Seznam použitých zkratek a symbolů

AJAX	- Asynchronní JavaScript a XML
ANOVA	- Analýza rozptylu
APU	- Accelerated Processing Unit
CSS	- Cascading Style Sheets
DOM	- Document Object Model
GPU	- Graphics Processing Unit
HTML	- Hypertext Markup Language
HTTP	- Hypertext Transfer Protocol
MD5	- Message-Digest Algorithm 5
MSSQL	- Microsoft SQL Server
MVC	- Model–view–controller
RIA	- Rich Internet Application
UI	- User interface
W3C	- World Wide Web Consortium
WCF	- Windows Communication Foundation
XHTML	- Extensible HyperText Markup Language
WPF	- Windows Presentation Foundation
XML	- Extensible Markup Language
XAML	- Extensible Application Markup Language
XUL	- XML User Interface Language

Obsah

Seznam obrázků

Seznam grafů

1. Úvod

1.1. Motivace

Mou hlavní motivací pro zvolení daného tématu bylo, že mě vždy zajímala témata uživatelských rozhraní. V dnešní době totiž na trhu často vítězí produkt ne proto, že má nejlepší funkce, ale proto, že se lépe ovládá a uživatelé z něho mají lepší pocit. Platí to ve všech oblastech, ne jen v oblasti softwaru a hardwaru. I přesto však spousta lidí význam uživatelských rozhraní a uživatelské zkušenosti podceňuje. A oblast bohatých internetových aplikací jsem si zvolil, protože si myslím, že je to směr, kterým se bude software v budoucnu čím dál tím více ubírat. Už dnes se všude spekuluje o budoucím masivním rozšíření cloud computeringu a bohaté internetové aplikace jsou jedním z prostředků, jak toho docílit. Vznikají již operační systémy, které jsou vesměs jen internetových prohlížečem a aplikace spouští z internetu. Proto jsem si chtěl v reálu ověřit, jak se RIA aplikace vytváří, jak se ovládají běžným uživatelům a které postupy ovládání jsou pro ně nejlepší.

1.2. Cíl práce

Cílem mé práce je zmapovat a ověřit možnosti tvorby bohatých internetových aplikací v prostředí internetu. Zjistit aktuální trendy a možnosti tvorby a implementace RIA aplikací a ověřit si, zda jsou uživatelé na RIA aplikace zvyklí a zda jim odlišná filozofie nečiní potíže. K tomu bude použita testovací aplikace, vytvořená v prostřední Microsoft Silverlight. Daná aplikace nastíní různé možné postupy při tvorbě RIA aplikací. Díky veřejnému testování reálných uživatelů bude poté vyhodnoceno, které postupy jsou pro ně ty nejlepší. Hlavním výstupem tedy bude vyhodnocení testování a nastínění nejlepší možné cesty tvorby RIA aplikací z pohledu ovládání a uživatelské přívětivosti.

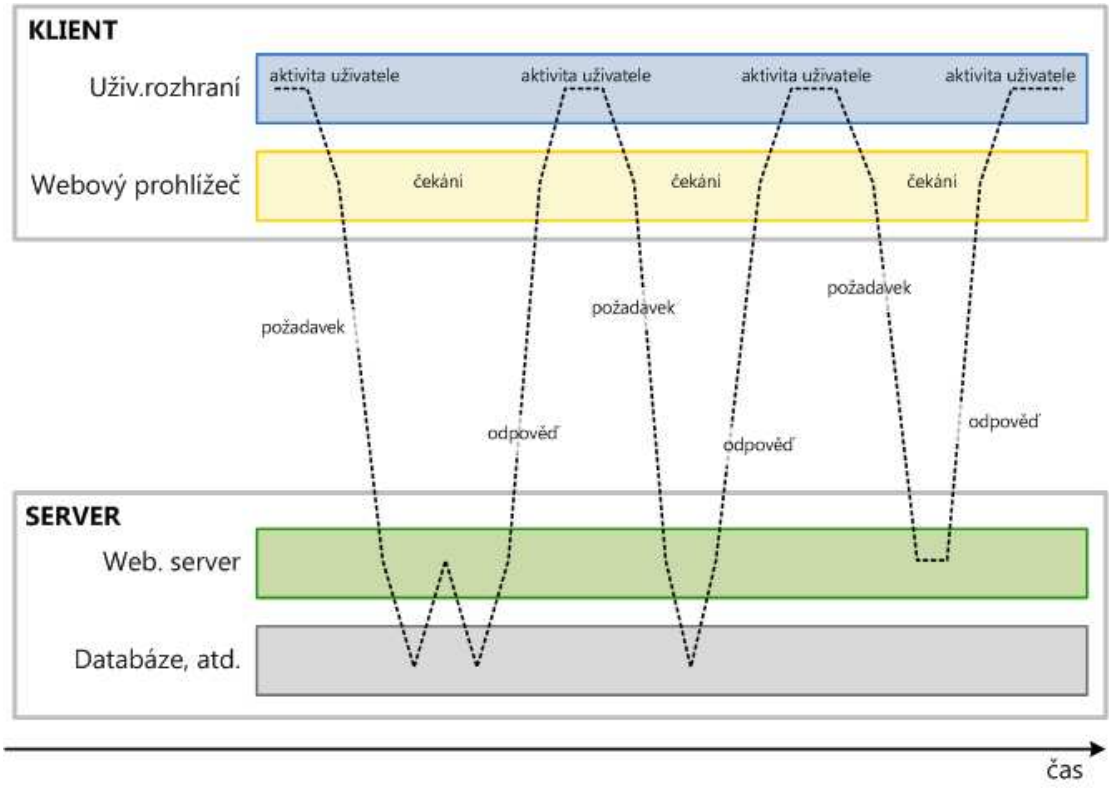
2. Bohaté internetové aplikace

Ze začátku by se hodilo si říct, co to vlastně zkratka RIA znamená. Zkratka RIA vychází z anglického Rich Internet Application, neboli bohatá internetová aplikace. Pojem Rich Internet Application není přesně definován, jedná se spíše o směr vývoje internetových aplikací. Samotná zkratka tak značí pouze jedinou aplikaci v jednotném čísle, proto když se bude mluvit o bohatých internetových aplikacích v množném čísle, bude používán pojem „RIA aplikace“. První zmínka o RIA aplikacích se datuje rokem 2002, kdy firma Macromedia (nyní součást Adobe) tento výraz použila pro popis výhod jejich nové verze produktu Macromedia Flash MX [1]. Samotný koncept však existoval už několik let předtím, např. technologie Remote Scripting firmy Microsoft z roku 1999.

Ve zkratce se dá říct, že se jedná o internetové aplikace, které přináší funkcionalitu a vlastnosti známé z desktopových aplikací. Většinou běží v internetovém prohlížeči bez potřeby instalace dalšího softwaru anebo případně jen plug-inu. Kvůli bezpečnosti většina RIA aplikací běží jen v chráněném prostředí s minimem práv, tzv. sandboxu (pískovišti). Nelze však s určitostí rozhodnout, která aplikace je anebo není RIA, protože často je pojem RIA přiřazován i aplikacím, které dané specifika nesplňují. O výrazu RIA se dá říct, že patří mezi tzv. Buzz word, (jako například pojem Web 2.0) a tak je někdy „zneužíván“ k marketingovým účelům.

Vraťme se však k pojmu bohatá internetová aplikace. Slovo „bohatá“ nám určuje, že internetové aplikace jsou více, než jen klasické stránky obsahující text a obrázky. Přináší nám funkcionalitu, na kterou jsem zvyklý z klasických desktopových aplikací, ať již jde o komplexnost grafického rozhraní, anebo komfort funkcí odpovídající klasickému desktopovému řešení, jako Drag and drop, klávesové zkratky, kontextová nápověda apod. V tradičních internetových aplikacích je interakce s uživatelem omezena na několik základních ovládacích prvků: checkboxy, radio buttony a formulářová pole. To často znemožňuje tvorbu snadno a plně použitelných aplikací. RIA aplikace však mohou využívat bohatší spektrum ovládacích prvků, které umožňují vyšší efektivitu a lepší komfort pro uživatele.

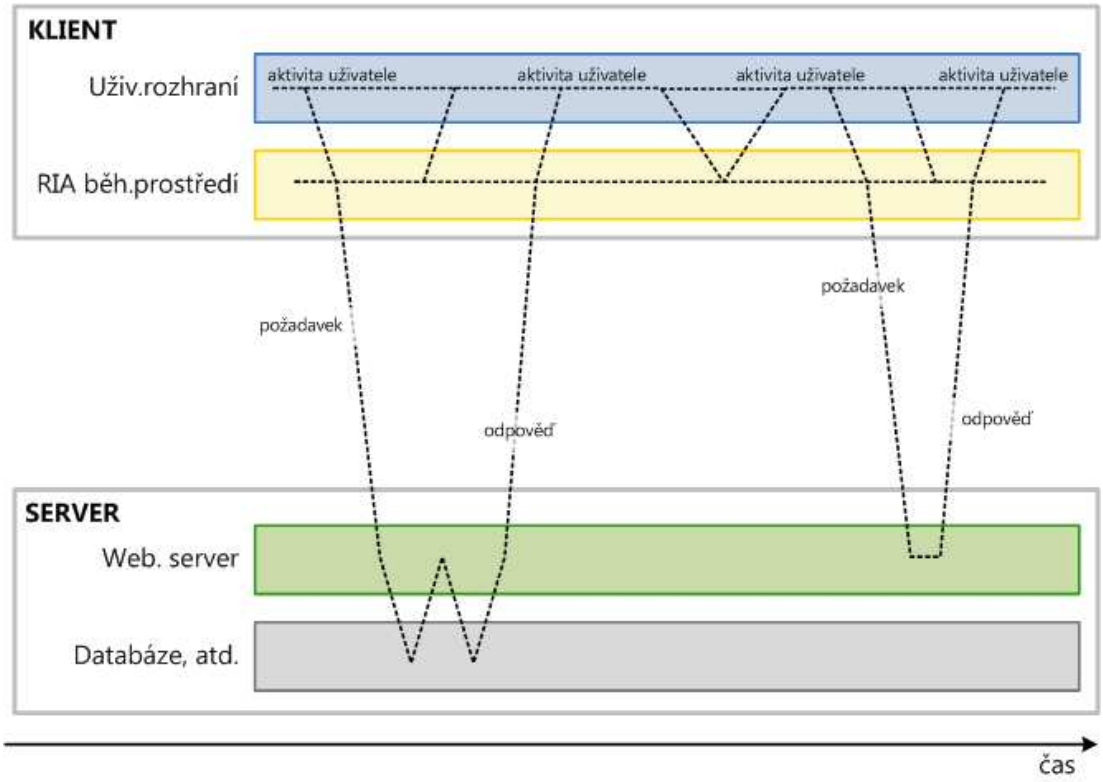
Mezi největší nevýhody klasických internetových aplikací patří omezení protokolu http. Ten vznikl pro výměnu hypertextových souborů v prostředí internetu. Funguje tedy na principu požadavek-odpověď, neboli synchronně viz obrázek 2-1 [2].



Obrázek 2-1: Synchronní přenos

Obrázek 2-1 popisuje klasickou internetovou aplikaci, například stránku napsanou pomocí HTML. Z obrázku jasně vyplývá, že pokud je na stránce potřeba změnit byť jen drobnost (validace, aktualizace části dat), musí se odeslat požadavek na server a vyčkat na odpověď. To nutí uživatele čekat, než jim server odešle opět novou stránku. A v době mezi odesláním požadavku a čekáním na odpověď s aplikací uživatel nemůže pracovat. To k uživatelské přívětivosti ani rychlosti práce s aplikací moc nepřispívá.

RIA aplikace naopak komunikují asynchronně, viz Obrázek 2-2. [2]



Obrázek 2-2: Asynchronní přenos

Aplikace sice stále využívá http, ale nekomunikuje se serverem přímo, nýbrž využívá běhové prostředí. V praxi to tedy znamená, že když je potřeba změnit něco na stránce, nemusí server odesílat celou stránku ale pouze pozměněnou část. A zároveň část požadavků nemusí být na server vůbec posílána, protože je vykoná přímo logika na straně klienta. Pro ještě lepší výkon obsahují RIA aplikace doplňkové technologie, jako real-time streamování, a lokální cacheovací mechanismy, které dokážou snížit dobu čekání a zvýšit rychlost odpovědi. Díky asynchronní komunikaci může většinou po odeslání požadavku uživatel s aplikací dále pracovat a nemusí čekat na odpověď.

Kromě vyjmenovaných výhod RIA aplikací oproti klasickým je ještě třeba zmínit výhodu vyššího výkonu. Zjednodušeně řečeno, pokud se RIA aplikace dokáže obejít bez neustálého posílání požadavků na server díky tomu, že jej bude zpracovávat na straně klienta, bude práce s ní určitě rychlejší. Zpracováváním na straně klienta se také zmenší zatížení serveru. Problémem však mohou být málo výkonná zařízení, jako například mobilní telefony.

2.1. Požadavky na moderní bohaté internetové aplikace

Jak již bylo řečeno, RIA aplikace nabízí funkčnost desktopových aplikací a často jsou nasazovány jako jejich náhrada. Zkusme si nyní nastínit ukázkový příklad, na kterém si ukážeme výhody, nevýhody a možnosti RIA aplikací.

Mějme softwarovou firmu. Pro svůj běh a správu používá širokou škálu programů. Například programy pro správu docházky, dovolených, projektů, zaměstnanců atd. Dané programy jsou

koncipovány jako desktopové aplikace, které komunikují s firemními servery. Pro jejich používání se musí uživatel přihlásit. Jelikož firma vyvíjí software pro více platforem, běží uživatelské stanice na různých platformách. Proto dané aplikace musí být vytvořeny pro různé platformy. Pokud je některý ze zaměstnanců na cestách a chce dané aplikace využít, musí je mít nainstalovány i na svém notebooku. Občas se požadavky na aplikace mění, a tak je potřeba update. Ten se musí poté vytvořit a otestovat pro všechny platformy. Přístup z mobilních zařízení není možný.

Co by se tedy stalo, pokud by firma přistoupila na řešení, že dané desktopové aplikace změní na bohaté internetové aplikace?

Předně by se na počítače nemusely instalovat žádné aplikace, stačil by systém s prohlížečem (případně plugin). V případě potřeby aktualizace aplikace by se nemuselo na cílových stanicích nic měnit, stačí jen aktualizovat aplikaci na serveru. Tím se notně sníží nároky na správu cílových stanic. Zároveň uživatelé vždy pracují s aktuálními aplikacemi a používají stejnou verzi. Velké výhody přinese i to, že se nebude muset aplikace vyvíjet pro více platforem. Aplikace by měla mít funkčnost na všech stanicích stejnou (mohou však nastat výjimky v závislosti na použité technologii).

Daný systém by se dal navrhnout jako centrální aplikace, kde se jen v závislosti na roli přihlášeného uživatele mění přístup k daným částem. Takže pokud třeba zaměstnanec povýší na roli vedoucího týmu, stačí jen v systému povolit přístup k novým částem aplikace. Nemusí se tedy nahrávat žádný nový software. Takhle řešený systém je poté přístupný i vzdáleně, například zaměstnancům na cestách anebo pracujícím doma. V závislosti na použité technologii je přístupný i pomocí mobilních zařízení.

Jelikož se data i samotná aplikace stahují ze serveru, potřebuje takové řešení pro svůj provoz mnohem méně diskového prostoru. To, že se data ani aplikace nenacházejí na počítači, přináší výhody i z pohledu bezpečnosti. Zároveň v případě pluginových technologií běží v tzv. sandboxu, a tak nemohou ani ohrozit cílový počítač. Složité výpočty může vykonávat server, a proto jsou i menší nároky na výkon pracovních stanic.

Jelikož by šlo o bohatou internetovou aplikaci, její ovládání i vzhled by mohl být stejný jako u desktopových aplikací.

Mezi nevýhody daného řešení patří větší datový provoz v sítí, jelikož se data ukládají na serveru a ne na cílové stanici. To samé platí i o nárocích na výkon serveru. Tyhle aspekty se však dají částečně omezit vhodným návrhem aplikace. RIA aplikace sice umožňují práci i v offline módu, bez připojení k síti, ale uživatel už musí mít stáhnutého klienta.

3. Technologie pro tvorbu bohatých internetových aplikací

Technologie pro tvorbu RIA aplikací by se daly rozdělit do dvou skupin. Zásadní rozdíl, který odlišuje tyto dvě skupiny, je ten, že jedna využívá technologie, které jsou dnes podporovány na úrovni samotných prohlížečů, bez nutnosti dodatečného softwaru. Do dané skupiny patří například AJAX nebo HTML5. Druhá skupina pro svůj běh potřebuje, aby byl na cílové stanici nahrán potřebný plug-in a nebo jiný dodatečný software. Do téhle skupiny spadá například Adobe Flex, Microsoft Silverlight nebo JavaFX.

Nepluginové **Pluginové**

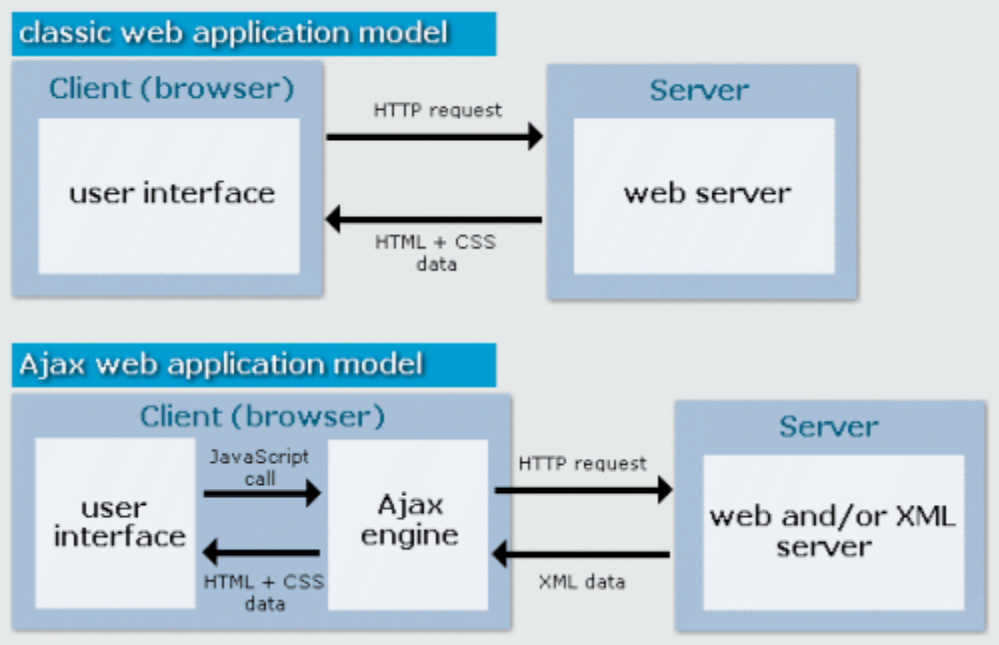
V následující části budou krátce představeny a poté porovnány z různých hledisek.

3.1. AJAX

Zkratka AJAX znamená Asynchronous JavaScript and XML. Předem by se hodilo říct, že AJAX sám o sobě není technologií ani softwarovým produktem. Jedná se spíše o koncept použití několika technologií dohromady s určitým cílem. Jedná se o technologie:

- HTML (nebo XHTML) a CSS pro prezentaci informací;
- DOM a JavaScript pro zobrazování a dynamické změny prezentovaných informací;
- XMLHttpRequest pro asynchronní výměnu dat s webovým serverem

Vše to jsou technologie, které podporuje samostatný prohlížeč, a jsou používány i při tvorbě klasických webových aplikací. To, co však dělá AJAX zajímavým, je spolupráce všech těchto technologií dohromady a především `XMLHttpRequest` pro komunikaci se serverem. Rozdíl proti klasickým webovým aplikacím je zobrazen na obrázku 3-1 [3].



Obrázek 3-1: Klasická webová aplikace a AJAX

V praxi to funguje tak, že pokud je potřeba na stránce něco změnit anebo odeslat informaci na server, nemusí se jako v případě klasické webové aplikace odeslat požadavek na server, počkat na vyřízení požadavku a poté čekat na znovunačtení celé stránky. To celé ještě synchronně. Při využití AJAXu se jen pomocí `XMLHttpRequest` odešle požadavek na server a server poté vrátí odpověď. Na výsledné stránce se pak překreslí jen změněná část. Zprávy jsou posílány asynchronně a většinou se pro výměnu zpráv používá formát XML.

Výhodou použití AJAXu tedy je odstranění potřeby znovunačítání celé stránky a s tím spojné zrychlení práce a uživatelské přívětivosti s danou aplikací. A jelikož není nutné při každém požadavku sestavit celou webovou stránku znovu, snižuje se tím i zatížení kladené na webové servery.

Vývojář však musí ovládat jak JavaScript, tak i CSS a HTML. Při vývoji musí tedy kombinovat všechny „jazyky“. Jde sice o technologie poměrně snadné, ale jak již bylo řečeno, jejich implementace je v různých prohlížečích různá. Proto se musí aplikace různě upravovat a ladit pro různé prohlížeče. Usnadnit vývoj mají různé frameworky jako například JQuery. Pokud budeme mluvit o vývojových prostředích, existují různé editory, které ale vesměs slouží jen jako pomůcka pro psaní kódu. Ladění a debutování probíhá přímo v prohlížečích. Zároveň JavaScript se podle mého názoru na vývoj větších a složitějších aplikaci nehodí. Vývoj obsáhlých aplikací pomocí něho je poměrně náročný a časově zdlouhavý, nehledě na nízký výkon takového řešení. Závěrem tedy vyplývá, že vývoj aplikací v AJAXU je poměrně pracný a navíc roztříštěný do více technologií. Zároveň dost zdlouhavý s nutností otestovat a vyladit kód ve všech prohlížečích a různých verzích.

.

3.2. HTML5

HTML5 je rozšíření jazyka HTML. Jde o novinku a v současnosti není ani plně definován. Zatím je jen ve stadiu návrhu organizací W3C [4]. Měl by však přinést nové zajímavé vlastnosti jako nové HTML tagy sémanticky definující strukturu stránky, perzistentní úložiště formou asociativního pole, relační databáze s podporou transakcí, podporu offline aplikací, vylepšenou podporu práce s multimédii a podobně. To vše bez nutnosti dodatečného softwaru, jen za pomoci prohlížeče.

Jelikož, jak už bylo řečeno, plná definice není ještě hotová, je i podpora v prohlížečích stále nízká. Podporují ho vesměs jen nejnovější verze prohlížečů, a to jen částečně. Zároveň, díky nehotové definici, každý ještě po svém. Pokud bychom například chtěli použít pokročilé HTML tagy video [5] a audio, neexistuje formát jak videa, tak audia, který by se mohl přehrát ve všech majoritních prohlížečích. [6]

HTML5 je určitě zajímavá technologie, která se v budoucnu nejspíše určitě prosadí. V dnešní době ji však nepovažujeme za plně použitelnou pro tvorbu RIA aplikací. To je taky důvod, proč se jí v dalších částech nebudeme zabývat.

Zároveň však díky tomu že AJAX není nijak striktně definován, dá se HTML5 považovat jako součást technologie AJAX. Proto se bude většina řečeného o AJAXu týkat taky i HTML5.

3.3. Adobe Flex

Adobe Flex je už zástupce plug-inových technologií a pro svůj běh potřebuje, aby byl v prohlížeči nainstalován Adobe Flash Player. Jde již o poměrně zaběhnutou technologii, první verze vznikla v roce 2004 tenkrát ještě ve společnosti Macromedia. V dnešní době tuto technologii vlastní a vyvíjí společnost Adobe. Laicky by šlo Flex přirovnat k technologii Flash, upravené pro potřeby programátorů. V praxi to tak částečně i je a věci jako běhové prostředí anebo programovací jazyk mají obě technologie společné. Teoreticky by šlo říct, že aplikace vytvořená ve Flashi se dá vytvořit i ve Flexi, a naopak. Hlavní důvod vzniku této technologie bylo však ulehčit vývojářům vývoj takové aplikace. Flash je přeci jen více zaměřen graficky, než na programování složitých aplikací.

Tvorba aplikací v Adobe Flex funguje tak, že uživatelská rozhraní se definují pomocí souboru MXML, což je druh XML souboru. Samostatná logika se píše v ActionScriptu, což je programovací jazyk kombinující rysy JavaScriptu a Javy. Pro samostatný vývoj slouží většinou Adobe Flex Builder, ale není to podmínkou. Výsledný kód se zkompiluje do souboru swf a poté nahraje na server. Prohlížeč si následně tento soubor stáhne a výsledný kód spustí v rámci stránky pomocí Flash Playeru.

V dnešní době se jedná o jednu z nejrozšířenějších technologií, což je dáno především vysokým rozšířením plug-inu Flash Player. Podle aktuálních statistik z března roku 2011 je až na 96% počítačů připojených k internetu nainstalován tento plug-in [7] .

3.4. Silverlight

Silverlight je další zástupce plug-inových technologií, tentokrát od firmy Microsoft. První verze, Silverlight 1.0, využíval pro aplikační logiku ještě jazyk JavaScript a celkově se s dnešními verzemi nedá srovnávat. Jako o první verzi blízkou k současného Silverlightu se dá považovat až verze 2.0. Ta byla uvedena v roce 2008 [8]. V parafrázi s technologií Flex, který vycházel z technologie Flash, Silverlight verze 2.0 vycházel z Windows Presentation Foundation (WPF). WPF je grafický subsystém pro generování grafického uživatelského rozhraní na platformě Windows. Jako součást .NET frameworku 3.0 byl poprvé uveden v roce 2006. I když samotný framework byl zaměřen na tvorbu aplikací desktopových, byla zde možnost spouštět aplikace i v internetových prohlížečích. Jako platforma pro RIA aplikace však nebyl příliš vhodný, protože ke svému běhu potřeboval .NET framework verze 3.0, jenž sice byl součástí Windows Vista, ale rozšířenost tohoto systému nebyla tehdy příliš vysoká. Proto by na ostatních systémech, Windows XP nevyjímaje, museli uživatelé stáhnout a nainstalovat .NET framework verze 3.0. Tím by se ale taky stal technologií, spustitelnou jen na operačních systémech Windows, a to až od verze XP. Proto byl v roce 2007 představen nový framework Silverlight (dříve označován Windows Presentation Foundation/Everywhere (WPF/E)) vycházející z WPF. Ten „chyby“ frameworku WPF pro běh v prohlížečích a tím spojený vývoj RIA aplikací odstranil. Uživatel si tak nemusí nahrávat kompletní objemný .NET framework, ale jen plug-in o velikosti 6Mb [9]. Tím byla i vyřešena, i když jen částečně, multi-platformnost tohoto řešení.

Uživatelská rozhraní se definují pomocí souboru XAML, což je jak u Flexu, druh XML souboru. Pro programování samostatné logiky může být použit jakýkoliv jazyk z .NET platformy. Nejpoužívanější jsou C# a Visual Basic. Jako vývojové prostředí slouží Visual Studio, je však možno použít i opensource variantu Eclipse4SL [10]. Výsledný kód se kompiluje do souboru XAP, který je zabalen pomocí ZIP. Výsledná aplikace se spouští v prohlížeči pomocí Silverlight plug-inu.

Technologie Silverlight je poměrně mladá, ale už si získala docela velkou popularitu u vývojářů. Výhodou je, že programátoři pro vývoj mohou používat svůj oblíbený jazyk z .NET platformy. Taky značka Microsoft má silné postavení a pravidelně vydává nové verze. Aktuální verze je 4.0, ale na konec roku 2011 je plánována verze 5.0. Podle aktuálních statistik z března roku 2011 je Silverlight plug-in nainstalován na 63% počítačů připojených k internetu [7].

3.5. JavaFX

JavaFX je platforma společnosti Sun Microsystems pro tvorbu RIA aplikací. Technologie JavaFX byla poprvé oznámena v roce 2007. První verze byla uvedena v roce 2008. JavaFX není cílena jen na vývoj webových aplikací, ale i na desktop a mobilní segment.

Jak už je z názvu patrné, JavaFX úzce vychází z Javy, a tak je do aplikace možné importovat přímo knihovny z Javy. Na rozdíl od ostatních představených technologií, aplikace je jen pomocí prohlížeče zavolána a poté běží v systému jako samostatný proces. Má však i více odlišností. Jak uživatelské rozhraní, tak i logika se zapisuje pomocí nového jazyka JavaFX script. Jde o skriptovací jazyk, který je postaven na platformě Java a syntaxí se podobá jazyku JavaScriptu. Jako vývojové prostředí slouží např. Netbeans a nebo Eclipse. Výsledný soubor se

kompiluje do souboru .fx . Samostatné spouštění aplikace probíhá tak, že pokud uživatel má nainstalovanou Javu SE 1.5, ale nemá běhové prostředí JavaFx, tak to se mu automaticky nainstaluje při spuštění programu. Velikost je v závislosti na aplikaci (podle toho, které knihovny používá) kolem 3Mb.

JavaFx je hodně mladá technologie a mezi její největší přednosti patří, že je široce podporována mobilními telefony. Stačí podpora Java ME a uživateli se poté stáhne běhové prostředí JavaFX. Podle aktuálních statistik z března roku 2011 je Java (tím není myšleno Java FX) nainstalována na 78% počítačů připojených k internetu [7]. Nevýhodou však je, že pokud uživatel nemá nainstalováno běhové prostředí Java, tak jeho následné stažení a naistalování je oproti jiným technologiím pracné a zdlouhavé. Taky reálných RIA aplikací vytvořených pomocí JavaFx je stále malý počet.

3.6. Ostatní technologie

Mezi ostatní technologie bychom mohli zmínit Curl, OpenLaszlo, Google Gears, XUL, Mozilla Prism atd. Jde však vesměs o technologie málo používané. Zmínit bychom mohli i Adobe Air, i když ten je zaměřen na desktopové aplikace, ale nabízí možnost tvorby webových aplikací (například jako WPF).

3.7. Krátké srovnání technologií

Je těžko určit které technologie je nejlepší. Pro různé úkoly se hodí různé technologie a hodně záleží i na preferencích a zkušenostech vývojáře.

3.7.1 Srovnání z pohledu rozšířenosti

Při porovnání, která z technologií je nejvíce rozšířená, by nejspíše vyhrál AJAX. Rozhodnutí tohohle problému však není tak snadné, jelikož u AJAXU nejde o jednu technologii, ale o spolupráci více technologiích (HTML,CSS,DOM a JavaScript). Ty jsou sice primárně podporované už samotnými internetovými prohlížeči bez nutnosti doinstalování dodatečného softwaru, ale každý prohlížeč má jejich implementaci různou. A tak určité vlastnosti JavaScriptu,HTML anebo CSS mohou mít implementované jinak anebo vůbec. Záleží tedy, které vlastnosti chceme použít. To je naopak výhoda pluginových technologií, kde když má uživatel nainstalován potřebný plugin, tak se výsledná aplikace zobrazí vždy stejně, nezávisle na prohlížeči.

U rozšíření ostatních technologií jsem vycházel z průzkumu z března roku 2011, který porovnává celosvětově rozšířenost pluginů v prohlížečích [7]. Nejrozšířenější z nich podle něj byl Adobe Flash Player s 96% (zdroj Adobe uvádí rozšířenost až 99% [12]). Daný plugin se používá pro zobrazování obsahu vytvořeného ve Flashi, Flexu a například i OpenLaszlo. Jako druhý nejrozšířenější byla Java s podílem 78%, používaná v RIA aplikacích pro zobrazení obsahu vytvořeného v JavaFX. Jako třetí byl Silverlight s podílem 62%. Zajímavostí je, že podíl Silverlightu z dlouhodobého hlediska stoupá, zatímco podíl ostatních technologií klesá. Podle jiných zdrojů [13] je podíl JavaFx a Silverlightu už shodný a to kolem 71%. Pokud odečteme verzi Javy 1.4 (kolem 2,5% z celkového podílu Javy), která nedokáže spouštět JavaFx kód, je Silverlight dokonce před ní.

3.7.2 Srovnání z pohledu multi-platformosti

Z pohledu multiplatformosti, spouštění aplikací na různých platformách, je na tom nejlépe AJAX a Flex. AJAX už svým pojetím není vázán na žádnou platformu a u Flexe existuje Adobe Flash plugin pro všechny významné platformy [14]. Silverlight má podporu pro platformu Windows (od Windows 2000) a Mac OS od verze 10.4. Existuje i verze silverlightu pro Linux zvaná Moonlight. Ta je však Microsoftem jen podporovaná a samostatný vývoj má na starosti firma Novel. Jeho implementace je však vzhledem k aktuálním verzím pozadu, Moonlight ve verzi Silverlightu 4 je stále jen v nekompletní preview verzi [15]. Nejhůře na tom z pohledu multi-platformosti je JavaFX, ta je podporována jen na platformě Windows (od Windows XP) a Mac OS od verze 10.5 [16]. Podpora pro Linux je stále jen v beta verzi a jediný podporovaný operační systém je Ubuntu 8.04 LTE.

Je však otázkou, nakolik je podpora pro operační systémy jiné jak Windows a MacOs významná. Podle statistik rozšíření operačních systémů Linux a ostatní systémy zabírají na trhu jen kolem 0,5-1% [17] (podle jiných zdrojů 1-2% [18]).

3.7.3 Srovnání z porovnání z pohledu výkonu

Při porovnání výkonu byla využita diplomová práce Tima Ernsta [19], v níž se zabývá porovnáváním výkonu jednotlivých technologií. Testovací úlohy byly: generování prvočísel, faktorizace prvočísel, dekódování JPEG, MD5 hashing, generování náhodného klíče, 2D test, 3D test a další. Při porovnání jeho výsledků a zároveň jeho závěrů se nedá s určitostí říct, která technologie je vyloženě nejrychlejší. Pokud bychom však měli vyvodit nějaké závěry, tak o AJAXu, respektive JavaScriptu se dá říct, že hodně závisí na prohlížeči. U některých testů byly rozdíly v rychlosti několikanásobně rozdílné i u stejných prohlížečů různých verzí. Taky u JavaScriptu nešlo implementovat 3D zobrazení, jelikož 3D API pro JavaScript ještě není oficiálně podporováno žádným prohlížečem. U ostatních pluginových technologií není tak snadné dojít k nějakému závěru. Například JavaFX excelovala v 3D zobrazení, ale naopak v generování prvočísel byly ostatní technologie i desetinásobně rychlejší. Ve většině případů pak byl nejrychlejší Silverlight (jen dvakrát byla těsně rychlejší JavaFX), ale naopak byl nejpomalejší, když šlo o grafiku 2D a 3D. Zajímavostí je, že verze Silverlightu pro Linux, MoonLight, byla v některých případech dokonce rychlejší než původní Silverlight. Autor práce se domnívá, že příčinou je nekompatibilita s použitým frameworkem. Flash (Flex) vykazoval vyrovnané výkony a hlavně exceloval v testech, které měly co dočinění s 2D grafikou. Zajímavostí je, že Flash jako pluginová technologie by měla být nezávislá na prohlížeči, a přitom zde byly patrné rozdíly při použití různých prohlížečů. Obecně se dá říct, že jsou úkoly, v kterých daná technologie exceluje a vzápětí na dalším selhává. Proto volba nejrychlejší technologie závisí hlavně na úloze jejího použití a taky způsobu implementace.

Na výkon aplikace má v dnešní době hodně vliv také to, jak dokáže využívat hardware na klientském zařízení. Jde hlavně o podpoře více vláknových aplikací a akcelerování pomocí GPU.

AJAX - defaultní podporu více vláknových aplikací nenabízí, akcelerace grafiky pomocí GPU však začínají implementovat některé prohlížeče.

FLEX – defaultní podporu více vláknových aplikací nenabízí [20] , nabízí však akceleraci grafiky pomocí GPU.

Silverlight – nabízí podporu jak pro tvorbu více vláknových aplikací, tak pro akceleraci pomocí GPU [21]

JavaFX - defaultní podporu více vláknových aplikací nenabízí [22]. GPU akcelerací to není tak snadné. JavaFX je plně GPU akcelerovaná, avšak zároveň není její podpora plně implementována.

Z tohoto přehledu tedy nejlépe vychází Silverlight, jelikož jako jediný nabízí jak podporu více vláknových aplikací, tak i akcelerace pomocí GPU. To je zvláště vhodné v dnešní době, kdy se používají procesory s více jádry a menší frekvencí a APU jednotky.

3.7.4 Srovnání z pohledu vývoje

V tomto srovnání byly hodnoceny technologie z pohledu snadnosti vývoje aplikací. Hodnocen byl jazyk, vývojové nástroje a další software, usnadňující vývoj dané aplikace.

U AJAXu, jak už bylo řečeno, nejde o jednu technologii, ale o soubor více technologií. Vývojář proto musí ovládat jak JavaScript, tak CSS a HTML. Při vývoji musí kombinovat všechny „jazyky“. Jde sice o technologie poměrně snadné, ale jak již bylo řečeno, jejich implementace je v různých prohlížečích různá. Proto se musí aplikace upravovat a ladit pro různé prohlížeče. Usnadnit vývoj mají různé frameworky jako například jQuery. Pokud budeme mluvit o vývojových prostředích, existují různé editory, které ale vesměs slouží jen jako pomůcka pro psaní kódu. Ladění a debugování probíhá přímo v prohlížečích. Zároveň JavaScript se podle mého názoru na vývoj větších a složitějších aplikaci nehodí. Vývoj takových aplikací by v něm poměrně náročný a časově zdlouhavý, nehledě na nízký výkon takového řešení. Podobný problém nastává i při tvorbě grafiky anebo uživatelského rozhraní jen pomocí HTML a CSS. Taková tvorba by byla složitá a zdlouhavá a platilo by u ní stejné omezení ohledně implementace v prohlížečích jako u JavaScriptu. Už z popisu tedy vyplývá, že vývoj aplikací v AJAXU je poměrně pracný a zdlouhavý, navíc roztříštěný do více technologií. Zároveň je dost pomalý s nutností otestovat a vyladit kód ve všech prohlížečích. Za Ajaxem nestojí taky žádná společnost, která by řídila jeho vývoj, ten je ponechán na komunitě. Potom je ale například problémem, že i když je různých frameworků pro tvorbu kolem 150, tak vesměs nejsou navzájem kompatibilní.

Flex jako jazyk používá ActionScript, což není zrovna rozšířený programovací jazyk. Vychází však z JavaScriptu, a proto by jeho naučení nemělo být pro vývojáře tak složité. Jako vývojové prostředí se používá Flex Builder.

U Silverlightu se pro vývoj může použít jakýkoliv jazyk z .NET frameworku, takže pro uživatele, kteří znají dané jazyky, je přechod na Silverlight snadnou záležitostí. Vývojové prostředí je taky známé Visual Studio, a tak se nemusí učit pracovat ani s novým nástrojem. Pro tvorbu grafické části programu se může použít Visual Studio, které od verze 2010 podporuje u Silverlight aplikací WYSIWYG editor, s možností editace grafických prvků bez nutnosti psaní kódu. Pro lepší práci s grafikou a tvorbu animací se používá Microsoft Blend, aktuálně ve verzi

4. Oba nástroje jsou vzájemně kompatibilní, a tak například může být projekt započat v Blednu, kde se vytvoří grafika, a pak se dá stejný projekt taktéž otevřít ve Visual Studiu. Přitom v obou nástrojích zůstane možnost vyvíjet jak grafiku, tak logiku dané aplikace. To usnadňuje spolupráci programátorů a grafiků.

JavaFX pro vývoj aplikací, jak již grafické části tak i logické části, používá JavaFX script, skriptovací jazyk, který je postaven na platformě Java a syntaxí se podobá jazyku JavaScriptu. Od nové verze 2.0 se již však nemá používat. Nahradit jej má nové API vycházející ze standardního jazyka JAVA [23]. To je podle mého lepší způsob, než nutit programátora se učit celý nový jazyk. Jako vývojové prostředí slouží např. Netbeans a nebo Eclipse. Pro vývoj grafických prvků se používá JavaFX Production Suite.

3.7.5 Srovnání z pohledu mobilního použití

Jeden z velkých požadavků a zároveň problémů pro RIA aplikace jsou mobilní zařízení. Mobilní zařízení, ať už jde o smartphony či tablety, jsou v dnešní době čím dál tím rozšířenější a dostupnější. Zároveň lze očekávat, že počet takových zařízení bude postupem času ještě růst. Většina těchto zařízení má přístup na web a jeden z hlavních účelů jejich užití je procházení internetu. Podle poslední statistiky [24] v severní Americe je už kolem 6% všech přístupů na web učiněno pomocí mobilního zařízení. Proto je nyní a hlavně pak do budoucna velmi důležité, aby byly RIA aplikace dostupné i pomocí těchto zařízení.

Takový přístup má však mnohé omezení, jako je velikost displeje, malý výkon zařízení a například různá specifika mobilních zařízení, např. nedostupnost pravého tlačítka, odlišné ovládání apod. Tím největší problémem je však omezenost internetových prohlížečů a samotných operačních systémů v takových zařízeních.

Pokud to opět vezmeme popořádku, tak největší podporu má tradičně AJAX technologie. Stále však nelze zaručit, že to, co funguje na klasickém desktopu, bude fungovat i na mobilních zařízeních.

Pokud jde o technologii Flash (využívá ji Flex), tak ta například není podporovaná v mobilních zařízeních společnosti Apple [25]. Android ho sice od verze 2.2 podporuje, ale pro jeho provoz je ještě potřeba určitý hardware [14]. Podpora Flashe pro Windows Phone 7, BlackBerry a ostatní platformy se zatím jen chystá [26]. Oficiálně je v dnešní době podporováno jen 33 zařízení [27].

Naopak Silverlight v této době nepodporuje žádný mobilní operační systém. Jediný systém, který ho má v blízké budoucnosti podporovat, je Windows Phone 7 [28].

JavaFX v dnešní době nemá žádnou podporu mobilních operačních systémů.

Jak je vidět, mobilní zařízení a RIA aplikace v dnešní době nejdou příliš dohromady. Lze jen spekulovat, zda je chyba v operační systémech anebo v RIA technologiích. Proto je vhodné u RIA aplikací vždy nabízet alespoň minimální alternativní obsah dostupný i skrze mobilní prohlížeče.

3.7.6 Srovnání pluginových a nepluginových technologií

Pokud jde o srovnání dříve definovaných pluginových a nativních technologií, obecně by se dalo říct, že pluginové technologie nabízí vyšší výkon a často i menší zatížení klientského zařízení. Také při tvorbě aplikací není vývojář omezen vlastnosti CSS a HTML. Rovněž se nemusí aplikace ladit pro každý prohlížeč a platformu zvlášť. Často nabízí i tzv. běh out-of browser, neboli běh aplikace i mimo prohlížeč. Pluginové technologie také mají často komplexnější a propracovanější vývojová prostředí. Jejich hlavní nevýhodou však je, že uživatel musí mít v prohlížeči nainstalován potřebný plug-in, a taky malá podpora pro mobilní zařízení.

4. Testování uživatelských schopností

Pro otestování a ověření určitých postupů a zásad při vytváření RIA aplikací a jejich uživatelských rozhraní byla vytvořena testovací aplikace, sloužící k uživatelskému testování. Jejím účelem bylo pomocí uživatelského testování otestovat určitá uživatelská rozhraní RIA aplikací pomocí automatických testů. Pomocí testů mělo být zjištěno, jak jsou uživatelé spokojeni s daným uživatelským rozhraním a jeho prvky, zda jim vyhovuje a nemají s jeho ovládáním nějaké problémy. Testy byly koncipovány tak, aby pokryly většinu klasických využívaných případů a scénářů u RIA aplikací. Celkově bylo vytvořeno 5 testů obsahujících několik testovacích případů. Cílem testů nebylo vytvořit co nejlepší uživatelské rozraní,ale otestovat rozhraní, se kterým se uživatelé u RIA aplikací potýkají anebo budou potýkat nejčastěji. V některých případech byly například do testů záměrně zanášeny „chyby“, aby se dalo ověřit, zda je daný postup doopravdy špatný.

4.1. Scénář průběhu testů

Scénář průběhu testů byl následující: Uživatel přišel na zvolenou stránku s testovací aplikací. Tam se mu zobrazily informace s vysvětlením účelu testu a popisem následných kroků. Důraz byl kladen na to, aby se uživatel necítil frustrován a aby věděl, že je testován program a ne on, což je častá chyba při uživatelském testování. Před začátkem testu bylo dáno uživateli na výběr, ať ohodnotí své schopnosti a zkušenosti s ovládáním internetových aplikací na stupnici 1-5 (5 je nejlepší). Po skončení mu bylo umožněno, aby mohl vyjádřit, co si o testu anebo daném UI myslí.

Z důvodu získání co největšího počtu uživatelských dat bylo vždy po skončení testu ještě uživateli navrženo, zda nechce zkusit jiné testy. Aplikace byla navržena tak, aby si pamatovala předešlé testy uživatele.

Samostatné testování bude probíhat různě v závislosti na typu testu a úkolu. V některých testech bude uživatel jen plnit testovací úkoly. Když si bude myslet, že vše splnil správně, tak klikne na `Pokračovat`. V jiných se uživatelovy kroky automaticky sledují a v závislosti na tom se mu ukazují nové úkoly. U každého testu byla nabídnuta možnost test předčasně ukončit z důvodu neporozumění zadání anebo neschopnosti vykonání daného úkolu. Pokud tester tuto možnost využil, byl mu následně nabídnut formulář, kde mohl vyjádřit, co mu v testu činilo potíže.

Každý test má svá specifika a testuje jiné části. Co však mají testy společné a co testuji u každého testu je:

Práce s myší – sledování polohy myši, uražené vzdálenosti a práci s kolečkem myši

Práce s klávesnicí – zachycování stisků všech kláves, klávesové zkratky a čas stisku

Časové údaje - měření čas trvání testu a časy mezi jednotlivými testovacími případy (u testů, které jsou takhle koncipovány).

4.2. TEST1 – Práce s formulářem

Test 1 měl ověřit práci s okny a formuláři. Tento test měl koncepci, že úkoly se zobrazovaly napravo v sloupečku a po splnění všech úkolů poté zvolil uživatel odeslání dat. Je to z důvodu toho, že se testuje například i procházení mezi poli, a tak kdyby uživatel klikal na potvrzující tlačítka, ztratil by „focus“ a nemohl by použít například klávesu `TAB`.

Seznam elementů, které byly ověřovány:

1.) Práce s okny. Při startu testu se objeví upozorňovací okno. Bylo zjišťováno, zda ho uživatelé zavřou křížkem, tlačítkem `Ok` případně `Cancel`, anebo zda ho nechají otevřené a jen posunou.
2.) Funkce kopírovat a vložit. Jde o jedny z nejčastějších úkonů při vyplňování různých formulářů, obzvláště pokud se některá pole opakují. Po testerovi bylo tedy v tomhle úkolu požadováno, ať do jednoho pole napíše libovolné město, a v následném poli ať ho zadá znova. Schválně nebylo přímo žádáno, ať text zkopíruje, ale jen mu byla nabídnuta situaci, která k tomu vybízí. Následně tedy bylo snímáno, zda použil zkratku `Ctrl+C` a `Ctrl+V` a nebo zda použil pro kopírování pravé tlačítko myši a kontextovou nabídku Kopírovat, Vložit, Vyjmout.
3.) Otestování rozbalovacího seznamu (ComboBoxu). Do rozbalovacího seznamu bylo umístěno velké množství zemí a tester měl za úkol vybrat Czech Republic. Zkoumalo se, zda výběr provedl tak, že danou položku v seznamu přímo vyhledal, nebo zda zadal několik úvodních písmen pro výběr.
4.) Otestování, zda testeři formulář odesílali pomocí klinutí na `Odeslat` a nebo stiskem klávesy `Enter`.
5.) Zda pro přepínání mezi poli formuláře používali klávesu `TAB` nebo klikali myší.
6.) Ve formuláři byla i položka s datem. Bylo v ní vyplněno aktuální datum a po testerovi se chtělo, ať vybere datum následujícího dne. U pole byl použit v dnešní době často používaný prvek DatePicker. Jde o malou ikonu, která po kliknutí zobrazí malý kalendář s možností přímé volby data. Bylo testováno, zda uživatel dal přednost tomu přepsat jednu číslici anebo si radši vybral datum z kalendáře.
7.) Další prvek, který byl otestován, bylo pole s číslem. Jako předvyplněné číslo bylo zvoleno číslo 1000 a po testerovi bylo požadováno, ať zadá číslo 997, neboli o 3 menší. U testovacího pole byla použita komponenta `NumericUpDown` neboli pole u kterého jsou dvě malé šipky nahoru a dolů. Slouží k výběru čísla bez použití klávesnice. Záměrně bylo požadováno číslo, které není tak vzdálené. Testuje se tedy, zda dá uživatel přednost zadání pomocí klávesnice anebo myši.
8.) Dále bylo připraveno pole, do kterého tester mohl vyplnit jakoukoliv hodnotu. Do pravé části však byla umístěna ikona s otazníkem. Po najetí na ikonu se objevil popisek se vzkazem, že uživatel má vyplnit hodnotu čísla 9. Podobná funkcionalita se často používá u formulářů, například proto, aby uživateli poskytla nápovědu k vyplnění daného pole. Tato vlastnost se často nazývá Tooltip.
9.) Celý formulář byl vložen do okna, které je menší, než je velikost formuláře. Uživatel tedy pro odeslání formuláře musí okno posunout. Následně se testovalo, zda pro posun používal posuvník, klávesy a nebo kolečko myši.

4.3. TEST 2 - TEST3 - TEST 4 – složité a obsáhlé aplikace

Testy 2, 3 a 4 byly vytvořeny pro otestování složitějších a obsáhlých aplikací. Takové aplikace mají složitá uživatelská rozhraní a pro nováčky, kteří s danou aplikací pracují poprvé, je práce s takovou aplikací náročná. Mezi takové aplikace například patří různé systémy pro účetnictví, správu zaměstnanců, skladů, zakázek apod. Jde však přímo o tu oblast aplikací, kde firmy často nasazují RIA aplikace na úkor desktopových aplikací. V takových případech tedy RIA aplikace nejvíce uplatní svoji „bohatost" a zároveň podobnost s desktopovými aplikacemi.

Testy byly postaveny tak, že přímo porovnávají 3 možné přístupy, při tvorbě uživatelského rozhraní pro takhle složitou aplikaci. Úkoly, které testeři vykonávali, tedy byli pro všechny 3 aplikace stejné.

Abych jsem se co nejvíce přiblížil reálným aplikacím, vybral jsem si, že má testovací aplikace bude simulovat účetní systém. Pro účetní systém, jako zástupce složitých RIA aplikací jsem se rozhodl proto, že jich je na trhu široké množství. A jelikož většina aplikací nabízí vesměs stejnou funkčnost a konkurence na tomto poli je vysoká, domnívám se, že o to více se tvůrci zaměřují na vzhled a ovládání takových aplikací. Většinou taky každoročně vychází nové verze, a tak je velká pravděpodobnost, že na nich budou nejlépe vidět nejnovější trendy v ovládání aplikací.

Pro vytvoření testů jsem proto prozkoumal uživatelská rozhraní a způsob ovládání většiny nejvýznamnějších účetních aplikací na českém trhu. Konkrétně šlo o software Pohoda, Money S3, Ekonom, Wind storm 10, Abra G2, Stereo 2007. Z nich jsem vybral klíčové prvky v jejich návrhu UI a ovládání.

Testy ovšem přímo nekopírují UI některého z daných systémů, to taky není ani předmětem mé práce, ale jen si z nich berou ty nejvýraznější prvky. To že jde o desktopové aplikace, není v tomhle případě na škodu, jelikož RIA aplikace je často mají simulovat a zastoupit. Mají však své určité specifika, jako tlačítko Zpět, Obnovení stránky, Adresní řádek apod. Po analýze daných systémů jsem je, podle způsobu jejich ovládání, rozdělil do 3 kategorií.

4.3.1 Test 2 – Klasické menu

První zástupce účetních systémů, obsahovalo klasické UI, které jsme mohli vidět např. u systémů Office 2003 a starší. Je to klasické a nejrozšířenější rozhraní pro ovládání podobných systému. Příkazy se vykonávají převážně přes hlavní menu a pod ním jsou ikony pro nejčastější funkce. Většina pozorovaných systémů používalo klasické menu, včetně na českém trhu asi nejúspěšnějším systémem Pohoda od firmy Stormware.

Seznam elementů, které byly ověřovány:

1.) Práce s datem. Od testu 1 se však mírně liší. V testu 1 bylo pole, které mělo přímo přednastavené datum. V tomhle případě však byla tři pole, která neměla přednastavené datum, ale byla jen zobrazena maska ve tvaru „DD.MM.RRRR“. Cílem bylo, stejně jako v prvním testu, otestovat, zda uživatelé dají přednost zadání data přímo, anebo ho vyberou z kalendáře. Zároveň bylo porovnáváno, zda to, že je datum přednastaveno a není zadáno jen maskou, zvětší šanci, že ho tester napíše rovnou, než že ho vybere z kalendáře. Například kvůli tomu, že mu stačí přepsat jedno číslo, a nemusí datum psát celé.
2.) Ověřovalo se, zda pro spuštění funkcí testeři raději použijí hlavní nabídku anebo ikony v panelu nástrojů pod ní. Teorie říká, že uživatelé u programů, které neznají a používají je prvně, dají přednost hlavní nabídce před ikonou [30]. Tímto testem se tak v praxi ověřovalo, zda je tomu doopravdy tak.
3.) Ověřeno to poté bylo ještě jedním testovacím případem. Testerovi se dalo za úkol spustit vybranou funkci pomocí ikony z panelu nástrojů. V následujícím úkolu po něm bylo žádáno spuštění další funkce. Ikona té funkce se však nachází přesně vedle ikony funkce předešlé. Je tedy velká šance, že při hledání první ikony narazil na tady tu, a dá tedy přednost spuštění dané funkce z panelu nástrojů před hlavní nabídkou.
4.) V testu bylo požadováno vyplnění polí jako variabilní a konstantní symbol, které měli obsahovat poměrně složitá čísla. Byl to úkol, který opět naváděl uživatele, ať daná čísla zkopírují ze zadání. Byly tedy jako v testu jedna sledovány klávesové zkratky, stisky pravého tlačítka myši anebo použití nabídky Kopírovat, Vložit z hlavního menu.

4.3.2 Test 3 – Postranní panel

Druhý zástupce účetních systémů by se dal definovat podobně jako ten předešlý. Jde taky o systém, který se ovládá přes hlavní menu. Rozdílem však je, že nalevo přibyl postranní panel, většinou ve formě rozklikávacího stromu (Ekonom, Money S3) anebo pomocí skrývajícího menu (Windstorm 10). Vzhledem k podobné funkcionalitě jsem dané přístupy spojil do jednoho testu a použil jen strom. Postranní panel vlevo v některých případech plní funkci panelu pro rychlý výběr funkce, a tím de facto nahrazuje panel nástrojů s ikonami pod hlavní nabídkou. Proto u některých zkoumaných aplikací tento panel úplně chybí. V testu však byl použit, jelikož má být zjištěno, zda je pro uživatele výběr často používaných funkcí vhodnější z panelu nástrojů s ikonami anebo z ovládacího stromového prvku.

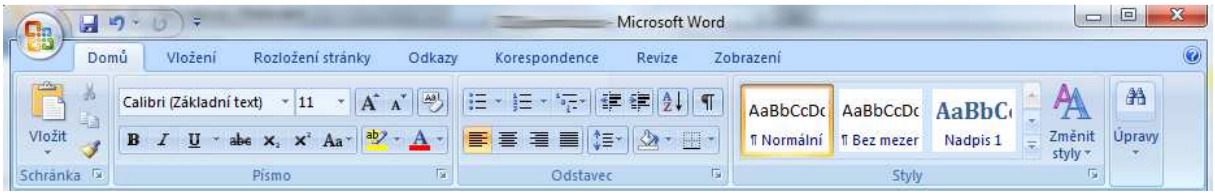
Seznam elementů, které byly ověřovány:

1.) Zda uživatelé dávali přednost stromovému postrannímu panelu před hlavním menu a panelem s ikonami.
2.) Celé levé stromové menu bylo umístěno v panelu, který se dá tzv. „připínat“. Uživatelé si pak mohou daný panel přemístit anebo schovat. Danou funkcionalitu lze nalézt v programu Visual Studio. Daná funkce se hlavně hodí pro uživatele, kteří s daným programem pracují často a vědí, které lišty potřebují a které ne. Proto si některou skryjí,

ať si zvětší svou pracovní plochu. Proto primárně nebylo očekáváno, že někdo danou funkcionalitu při prvním seznámení použije. Byla zde umístěna spíše proto, aby ověřila, kolik uživatelů si daný prvek alespoň vyzkouší.

4.3.3 Test 4 - Ribbon

Poslední třetí zástupce používá ke svému ovládání prvek Ribbon, neboli pás karet, viz Obrázek 4-1: Příklad rozhraní Ribbon. Jde o druh menu, kde jsou horizontálně rozprostřeny karty. Uživatel kliknutím na ouško vybere danou kartu, která sdružuje podobnou funkcionalitu. Samostatné prvky jsou ještě v rámci záložky sdružovány do skupin. Proto například pro vložení záhlaví stačí vybrat záložku „Vložit“ a v ní skupinu „Záhlaví a zápatí“ a zvolit „Záhlaví“. Ribbon byl poprvé uveden v Microsoft Office 2007. V dnešní době jej Microsoft postupně přidává do dalších nových aplikací.



Obrázek 4-1: Příklad rozhraní Ribbon

Bohužel jediný účetní software využívající Ribbon jsem našel jen u společnosti Ježek software. Byl použit u nové verze účetního programu DUEL 7 2011. Více aplikací jsem s daným rozhraním nenašel. Je to dáno buď tím, že se společnosti bojí, že jejich uživatelé jsou příliš konzervativní, a proto nechtějí výraznější změny ovládání anebo tím, že se dané rozhraní pro takové produkty nehodí.

Seznam elementů, které byly ověřovány:

1.) Zajímavostí Ribbonu je prvek nazvaný Quick Access menu. Nachází se vedle hlavního tlačítka a nabízí rychlý přístup k nejzákladnějším funkcím. V našem testu mu byly přiřazeny funkce `Uložit`, `Zpět` a `Tisk`. Cílem bylo tedy pozorovat, jak často uživatelé tento prvek používají a zda mu dávají přednost před klasickým menu.
2.) Ribbon stále obsahuje jeden prvek, který se podobá klasickému menu. Jde o souborové menu, kontextovou nabídku, která se objeví po kliknutí na kruhové tlačítko (v nových verzích Office nahrazeno podlouhlým barevným tlačítkem). Nabídka obsahuje vesměs stejný obsah jako u klasického menu, většinou po kliknutí na položku „Soubor“ v hlavním menu. Bylo tedy pozorováno, zda testeři pro výběr např. `Tisku` dávali přednost Souborovému menu, Quick Acces menu anebo funkci vyberou z panelu Ribbon menu.

4.4. Test 5 – Práce s tabulkou

V posledním pátém testu se testovala práce s tabulkami. Tabulky jsou důležitou součástí většiny firemních RIA aplikací. Samy o sobě však na první pohled působí jednoduše, protože nemají skoro žádné ovládací prvky. Mají však určité standardy chování, tím je myšlen např. multi select pomocí držení `Ctrl` anebo `Shift` apod. Proto taky jeden z cílů bylo otestovat, jak jsou uživatelé na tyhle vlastnosti zvyklí a zda jsou všeobecně známy.

Seznam elementů, které byly ověřovány:

1.) Bylo zkoumáno, zda uživatelé pro výběr více prvků používají držení klávesy `Ctrl` případně `Shift` a kliku myší.
2.) Další chování, které bylo testováno, je seřazení prvků po kliknutí na záhlaví daného sloupce. Jde o jednu ze základních funkčností, která však není nikde popsána. Proto bylo sledováno, kolik testerů ji využije.
3.) Smazaní záznamu, zda tester použil klávesu `Delete` a nebo tlačítko `Smazat`.
4.) Filtrování. V záhlaví každého sloupce byla funkcionalita, která umožňovala pokročilé filtrování jako: záznamy obsahující, končící na, začínající na apod. Záměrem opět bylo ověřit si, kolik testerů danou funkcionalitu vyzkouší a jaké filtry použijí.
5.) Funkce „drag and drop“, neboli česky táhni a pusť, je často vzpomínána při definicích bohatosti RIA aplikací. Jde o parafrázi reálného života a pro uživatele by to tedy měl být nejpřirozenější způsob při přemísťování dat. Úkolem pro testera tedy bylo přesunout data z tabulky nalevo do tabulky napravo. Pro přesun buď mohl využít tlačítko Přesunout anebo přesun mohl ještě vykonat pomocí Drag and Drop. Další z variant bylo využít klávesové zkratky `Ctrl+X` a `Ctrl+V`. Bylo tedy sledováno, kterému z těchto přístupů dal tester přednost.
6.) Ještě jednou pro ověření práce s Drag and Drop bylo dát testerovi za úkol přímo „přetáhnout“ prvek z levé tabulky do pravé tabulky. Bylo tak testováno, zda si pod slovem přetáhnout představí Drag and Drop anebo opět jen klikne na tlačítko Přesunout.
7.) Jako u jiných testů i zde byla sledována práce s kolečkem. Test byl koncipován tak, že i při vysokých rozlišeních se všechna data do okna nešla a musely se testerovi objevit skrolovací lišty.

5. Testovací aplikace

Testovací aplikace byla naimplementována pomocí technologie Silverlight. K implementaci byl použit Silverlight 4 [29] pro klientskou část a WCF (Windows Communication Foundation) pro část serverovou. Jako programovací jazyk byl zvolen C#. Výsledná data byla ukládána do MSSQL databáze, jako dotazovací jazyk byl použit LINQ. Při tvorbě byly použity komponenty firmy Telerik. Firma Telerik je jedna z největších firem pro tvorbu komponentů pro software firmy Microsoft, ať už jde o ASP.NET, WPF, Windows Form, Windows Phone a hlavně pro Silverlight, kde jejich komponenty patří mezi nepoužívanější. Nabízí k nim také širokou podporu, jako rozsáhlou dokumentaci, vzorové příklady a hojně navštěvované fórum. Konkrétně byla použita trial verze RadControl Q2 2010 SP2 [29]. Ta má funkcionalitu plné verze, jen při spuštění aplikace obsahující trial komponenty se zobrazí upozornění, že jde pouze o trial verzi. Jako vývojové nástroje byly použity Visual Studio 2010 a Microsoft Blend 4 [31].

Architektura aplikace byla MVC, neboli model-view-controller. Aplikace byla navržena tak, aby otestovala a ověřila všechny výše uvedené pokyny a domněnky. Zajímavostí například bylo naimplementování funkce kopírování po stisku pravého tlačítka myši. Většinou jsou totiž uživatelé zvyklí při práci jak s Flashem tak i Silverlightem, že při stisku pravého tlačítka se zobrazí jen nabídka s nastavením daného pluginu. Zároveň pokud jde o kopírování, Silverlight sice povoluje práci se schránkou, ale kvůli bezpečnosti se vždy uživatele zeptá, zda může mít daná aplikace k těm datům přístup.

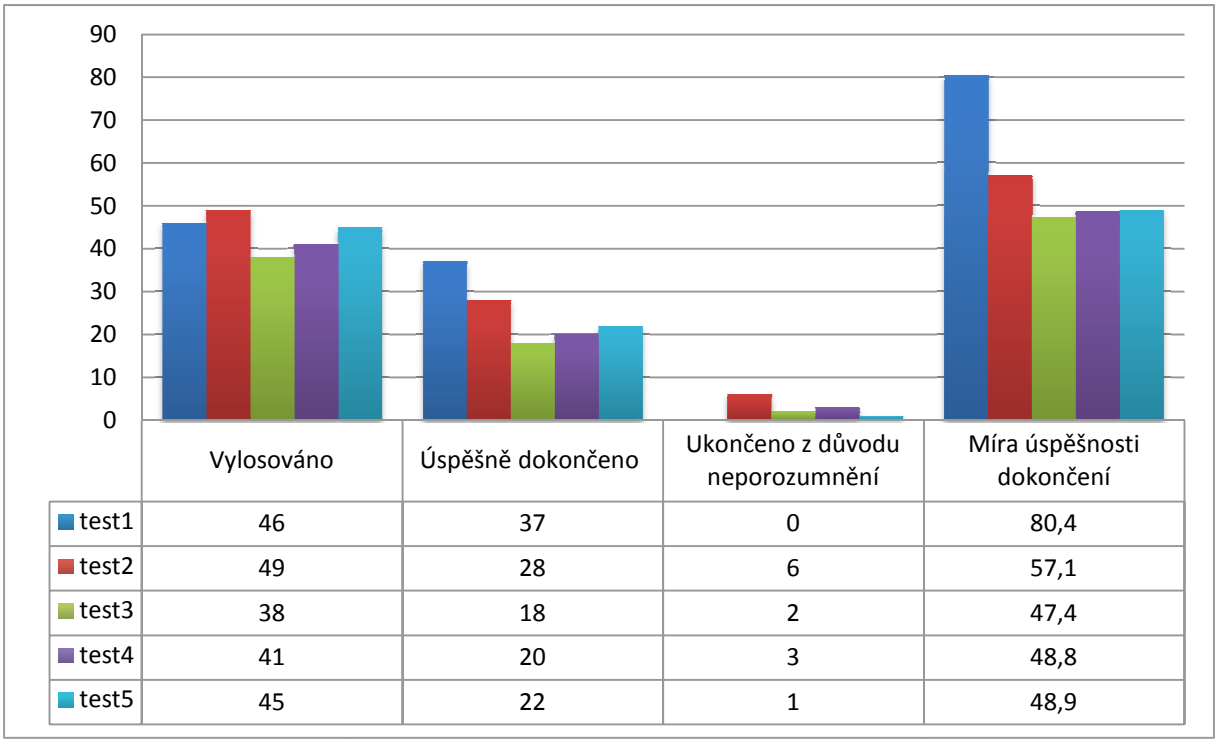
Zajímavé je i použití kolečka pro skrolování a to hlavně z toho důvodu, že Flash tuto funkčnost defaultně nepodporuje. Samozřejmě se dá daná funkcionalita doprogramovat, ale spousta vývojářů na to zapomíná. Výsledkem je, že kolečko pro skrolování ve většině aplikací nefunguje, a skroluje se jím celá stránka, ne daný prvek ve flashové aplikaci. Silverlight však skrolování pomocí kolečka podporuje defaultně.

Pro uchování informací o vykonaných testech a identifikaci testera se využívalo `Isolated Storage`, což je v Silverlightu obdoba webových cookies. Jelikož typ testu, který daný tester dostane, byl vždy losován náhodně, díky `Isolated Storage` se dalo předejít tomu, aby mu byl vylosován test, který už někdy v minulosti vykonal. Informace v `Isolated Storage` se ukládají pro každý uživatelský profil zvlášť, ale pro případ, že by byl profil sdílen mezi více uživateli, byla nabídnuta i možnost smazání těchto uložených dat.

Jak aplikace výsledně vypadala, lze vidět ve vyšším rozlišení v příloze na snímcích A – K.

6. Vyhodnocení testování

Jak již bylo řečeno, aplikace byla volně přístupná testerům po dobu jednoho měsíce. Celkově bylo tedy spuštěno 219 testů, úspěšně ale bylo dokončeno jen 127 z nich. Úspěšně dokončeno se myslí stav, kdy uživatel prošel celý test až do konce a došlo k odeslání výsledku. Z celkového počtu úspěšně dokončených testů byly už na první pohled 2 vyloučeny, protože byly vykonány během několika sekund. Takže výsledky se dále vyhodnocovaly jen ze 125 testů. Volba přidělování testů byla automatická. Nejvíce byl přidělován test 2 – Klasické menu. Průměrně každý vylosovaný test byl v 50% případů nedokončen. Jen test číslo 1 – Práce s formuláři byl dokončen v 80% případů. Je to dáno nejspíše tím, že byl ze všech testů nejkratší a nejlehčí na vykonání. Odpovídalo by tomu i to, že tenhle test ani jeden tester neukončil z důvodu neporozumění zadání.



	Vylosováno	Úspěšně dokončeno	Ukončeno z důvodu neporozumnění	Míra úspěšnosti dokončení
test1	46	37	0	80,4
test2	49	28	6	57,1
test3	38	18	2	47,4
test4	41	20	3	48,8
test5	45	22	1	48,9

Graf 1: Statistika testů

Aplikace byla propagována jak na titulní stránce katedry informatiky, tak i na různých počítačových fórech. Zde je taky možno brát vysvětlení, proč tak velké procento testerů uvedlo, že jsou zkušení anebo velmi zkušení v oblasti internetových aplikací.

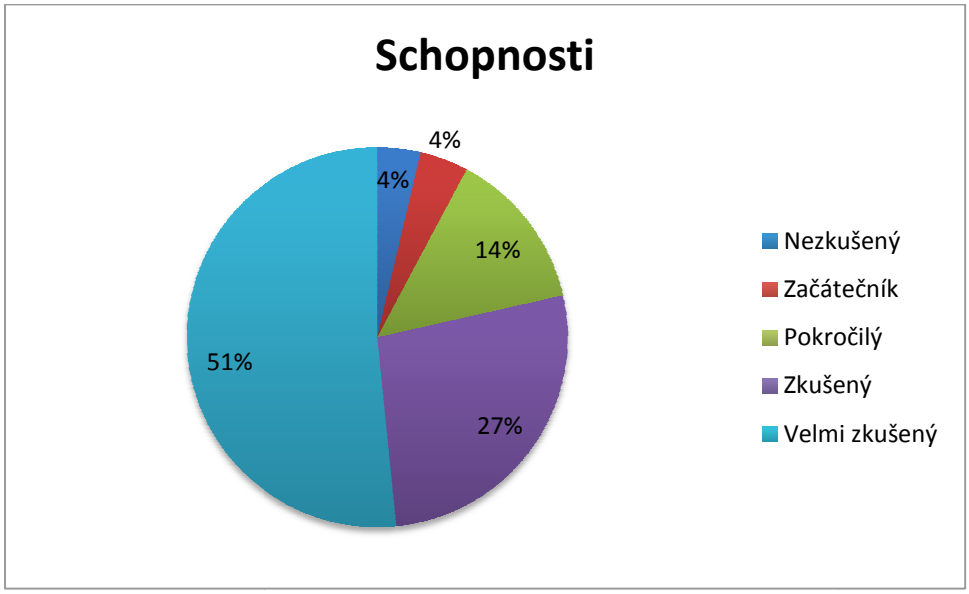
Samozřejmě jako u všech průzkumů (výběrových šetření) vycházejících ze vzorku populace, nelze závěry jimi dosažené brát jako dogma. Čím je však zkoumaný vzorek populace větší, tím přesnější závěry vzhledem k celkové populaci je možno získat. Bohužel získat potřebný vzorek testerů je věc poměrně náročná. Proto se to v praxi provádí tak, že se najmou za finanční odměnu lidé z různých oblastí, vzdělání, věku a jim se nechají vyplnit testy. V našem případě se však kvůli nedostatku prostředků dala přednost dobrovolné účasti náhodných testerů. Samotné testování je však činnost poměrně časově náročná. Proto se testů vesměs zúčastnili jen

uživatelé, kteří mají o podobnou oblast užší zájem. To je také důvod, proč v našem případě není vzorek tak obsáhlý a ještě je zkreslen tím, že většinu testů vykonávali testeři v daných aplikacích zkušení či velmi zkušení.

6.1. Schopnosti testerů

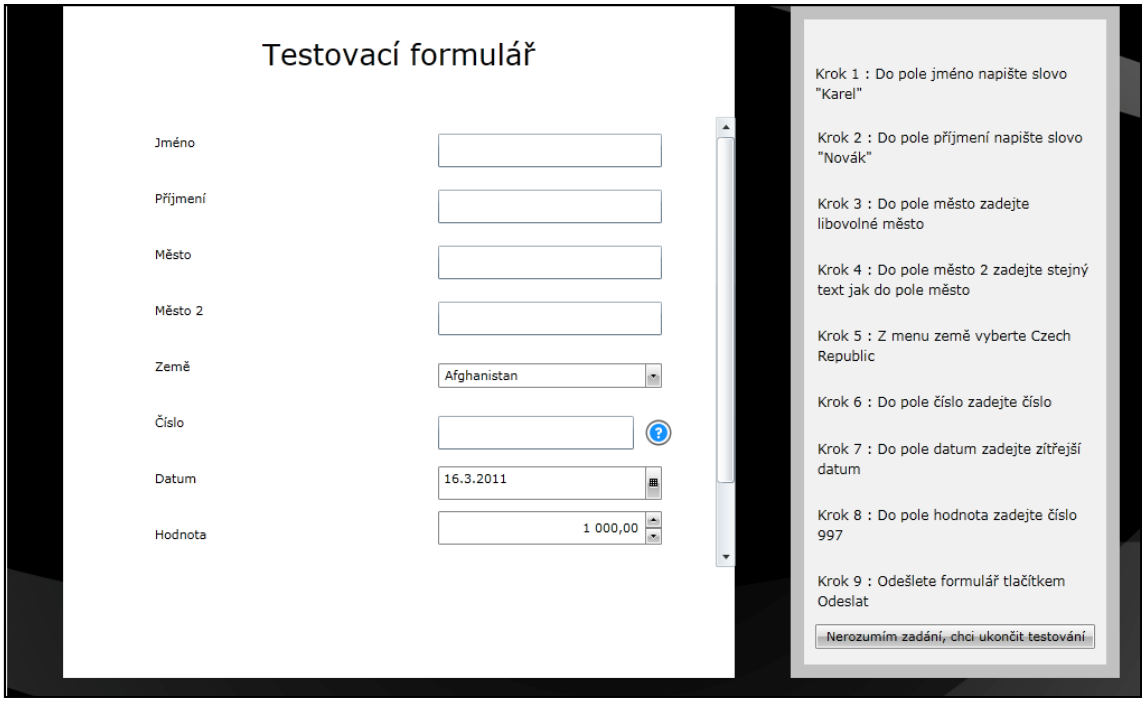
Nejvíce testerů uvedlo, že jsou Velmi zkušení z pohledu zkušeností s internetovými aplikacemi. I druhá nejpočetnější skupina uvedla, že je Zkušená. To by odpovídalo předpokladu, že testy nejvíce vykonávali odborníci a studenti informatiky.

Nezkušený	Začátečník	Pokročilý	Zkušený	Velmi zkušený
8	9	30	59	113



Graf 2: Schopnosti testerů

6.2. TEST 1 – Práce s formulářem

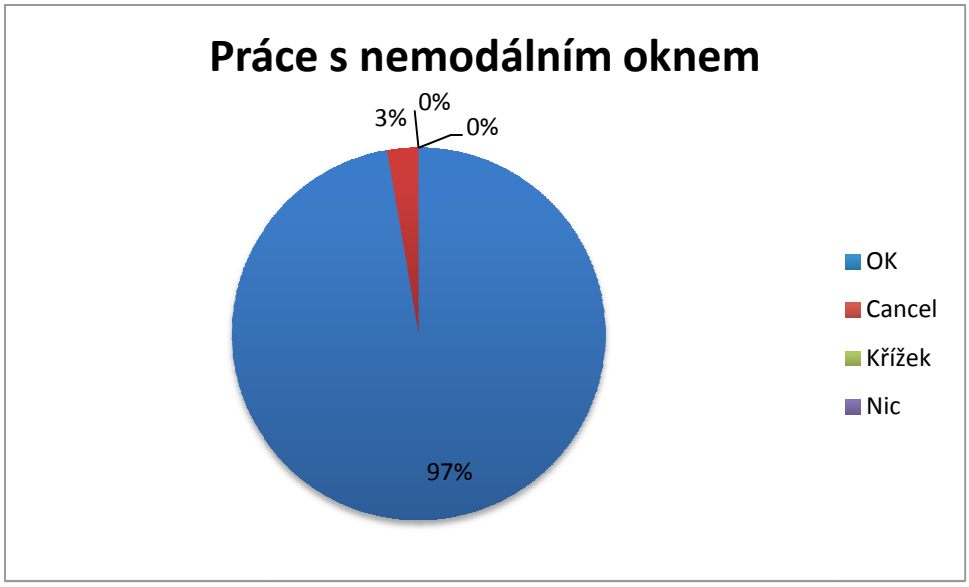


Obrázek 6-1: Test 1 - Formulář - ukázka obrazovky

6.2.1 Práce s nemodálním oknem

Testerovi se při spuštění testu zobrazilo oznamovací okno. Mohl ho poté zavřít kliknutím na tlačítko `OK`, `Cancel` a nebo pomocí křížku v rohu okna. Další z možností bylo odsunout ho na stranu a nedělat s ním nic. Z výsledků vyplývá, že 97% všech testerů pro zavření použilo tlačítko `OK`.

	Absolutní četnost	Relativní četnost
OK	36	97%
Cancel	1	3%
Křížek	0	0%
Nic	0	0%

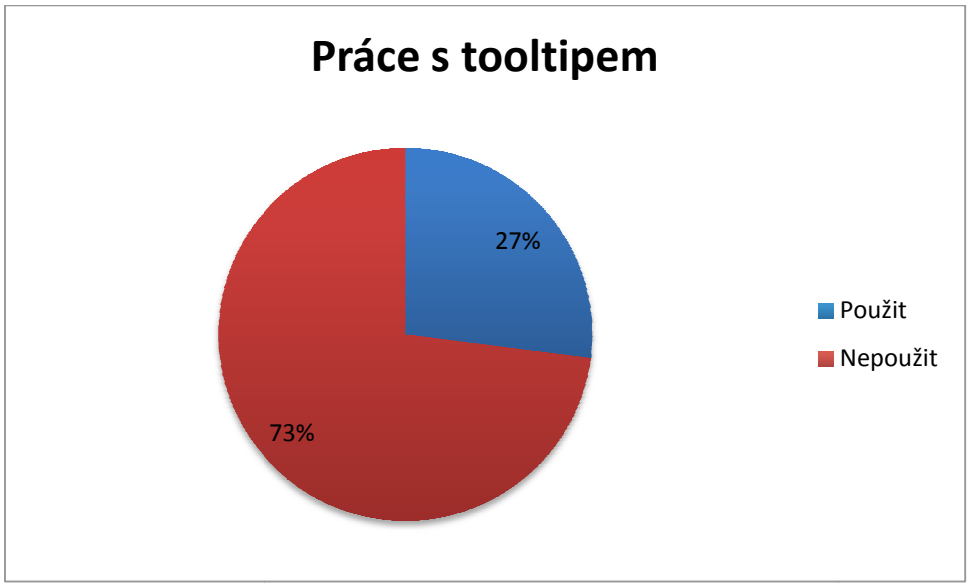


Graf 3: Práce s nemodálním oknem

6.2.2 Práce s Tooltipem

Ve formuláři bylo zobrazeno pole a nebylo předem určeno, co do něj má uživatel doplnit. Vedle pole však byla zobrazena malá ikona otazníku. Když na něj uživatel najel myší, ukázal se nápis „Doplňte číslo 9“. Byla tedy zkoumána práce s tzv. „Tooltipem“. Avšak jen 27% všech testerů si toho všimlo a vyplnilo číslo 9. Proto není vhodné umísťovat důležitě pokyny jen pomocí Tooltipu.

	Absolutní četnost	Relativní četnost
Použit	10	27%
Nepoužit	27	73%

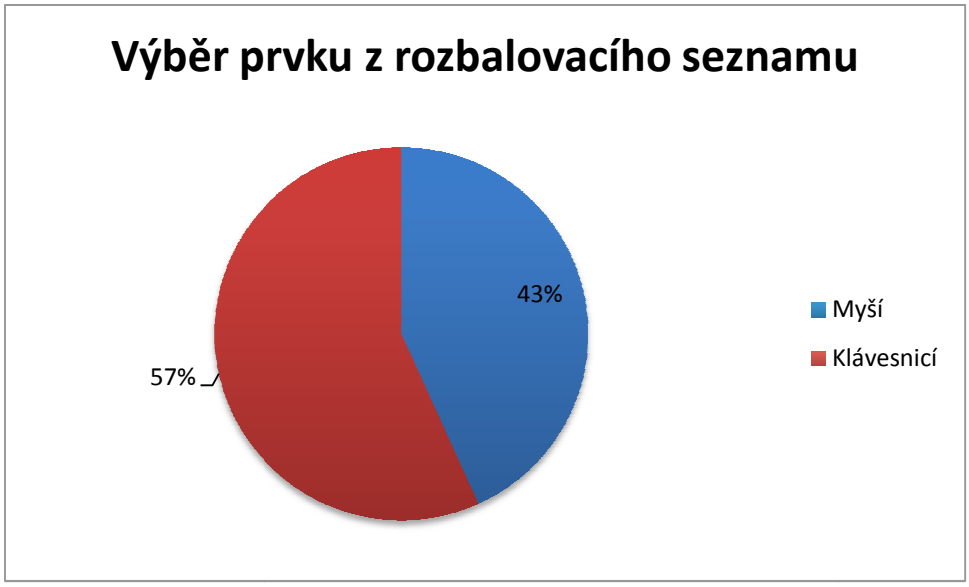


Graf 4: Práce s Toolipem

6.2.3 Výběr prvku z rozbalovacího seznamu (Combo box)

Ve formuláři byl obsažen rozbalovací seznam (Combo box) obsahující země a tester měl vybrat Czech Republic. To mohl buď pomocí myši anebo napsáním pár prvních písmen. Jelikož seznam zemí byl poměrně obsáhlý, výběr pomocí myši by byl pomalejší než pomocí napsání prvních pár písmen. Pole bylo uděláno jako automaticky doplňující, takže stačilo napsat písmena `C` a `Z` a automaticky se vybrala země Czech Republic. Pomocí klávesnice tedy rozbalovací seznam ovládalo 57% uživatelů, a proto je vhodné funkci výběru pomocí napsání prvních pár písmen implementovat, uživatelé jsou na ni zvyklí

	Absolutní četnost	Relativní četnost
Myš	16	43%
Klávesnice	21	57%

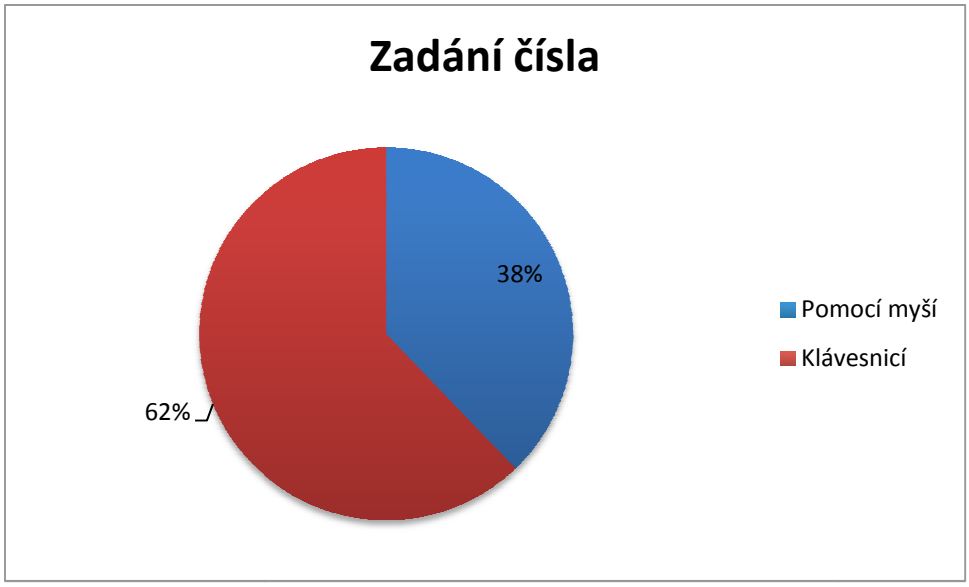


Graf 5: Výběr prvku z rozbalovacího seznamu

6.2.4 Zadání čísla

V tomto případě se testovalo, zda pro zadání čísla blízkému číslu zobrazenému využijí testeři toho, že původní číslo smažou a nové zadají napřímo klávesnicí nebo využijí šipek u komponenty `NumericUpDown`. Zadání pomocí šipek použilo 62% uživatelů. Pokud je tedy očekáváno, že uživatelé budou zadávat čísla blízká, je vhodné implementovat tuto funkčnost. Otázkou však je, kde je hranice, kdy je číslo moc vzdálené na to, aby ho zadali pomocí myší.

	Absolutní četnost	Relativní četnost
Myš	14	38%
Klávesnice	23	62%



Graf 6: Zadání čísla

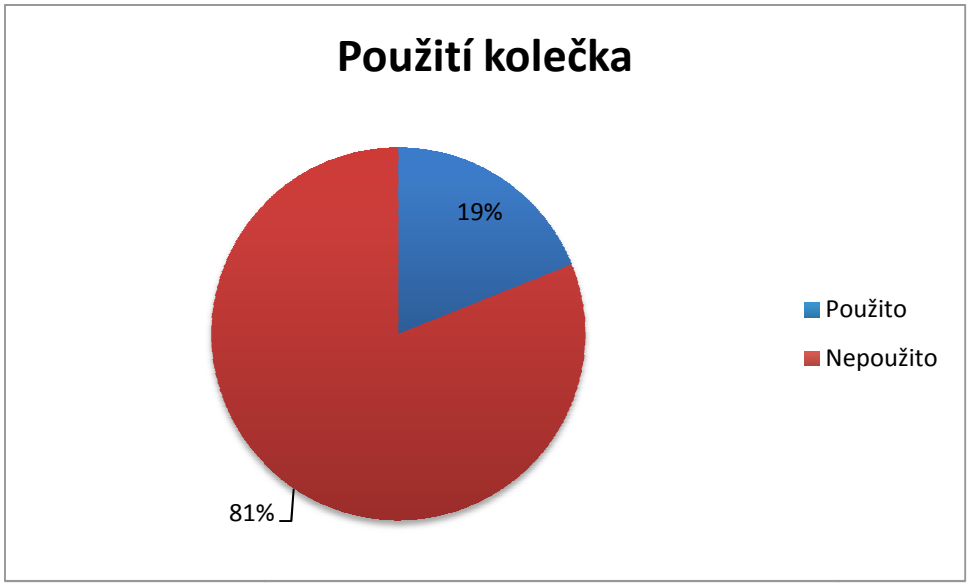
6.2.5 Kopírování

Testeři měli do pole zadat libovolné město a do dalšího pole ho zadat znova. Vybízelo to tedy k použití kopírování. Z celkového počtu 37 testerů však využili tuhle možnost jen 3 z nich a pro kopírování použili klávesovou zkratku. V tomhle případě je tedy vidět, že pro většinu uživatelů bylo snazší údaj zapsat ručně, než jej kopírovat.

6.2.6 Použití kolečka

Celý formulář byl vložen do okna, které je menší, než je velikost formuláře. Tester tedy pro odeslání formuláře musel okno posunout. Ve dvou případech pro posunutí bylo použito klávesy `Home`. Klávesa `Page Down` nebyla použita ani jednou. Pokud však jde o použití kolečka, překvapivě bylo použito jen v 19% případů. Přesný důvod tak nízkého počtu můžeme jen hádat. Svou roli v tom může hrát už jednou zmiňovaný problém Flash aplikací anebo to, že v posledním poli uživatelé stiskli opět klávesu TAB, a tak je to přesunulo na tlačítko Odeslat. Při zkoumání se však zjistilo, že přesunutí na tlačítko odeslat použili jen 2 uživatelé.

	Absolutní četnost	Relativní četnost
Použito	7	19%
Nepoužito	30	81%

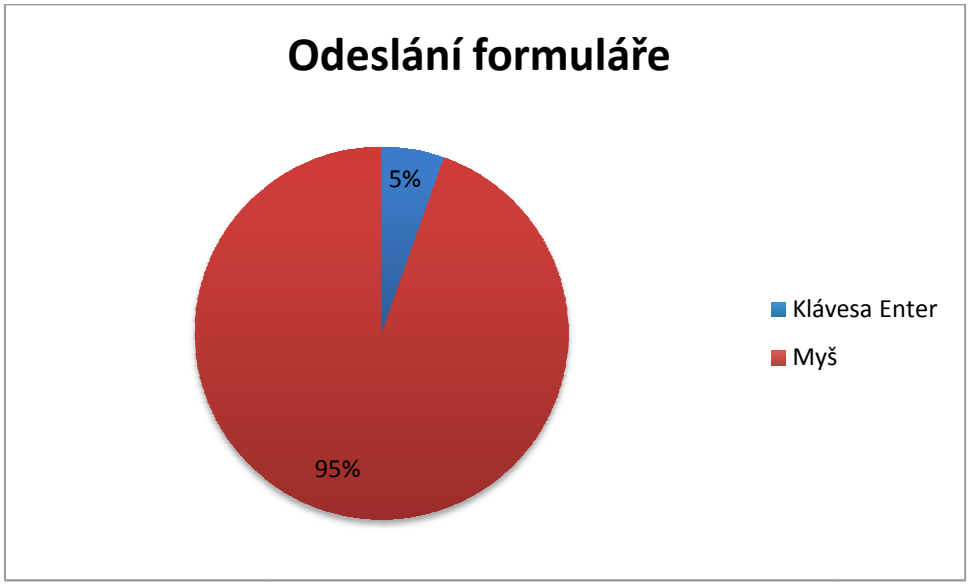


Graf 7: Použití kolečka

6.2.7 Odeslání formuláře

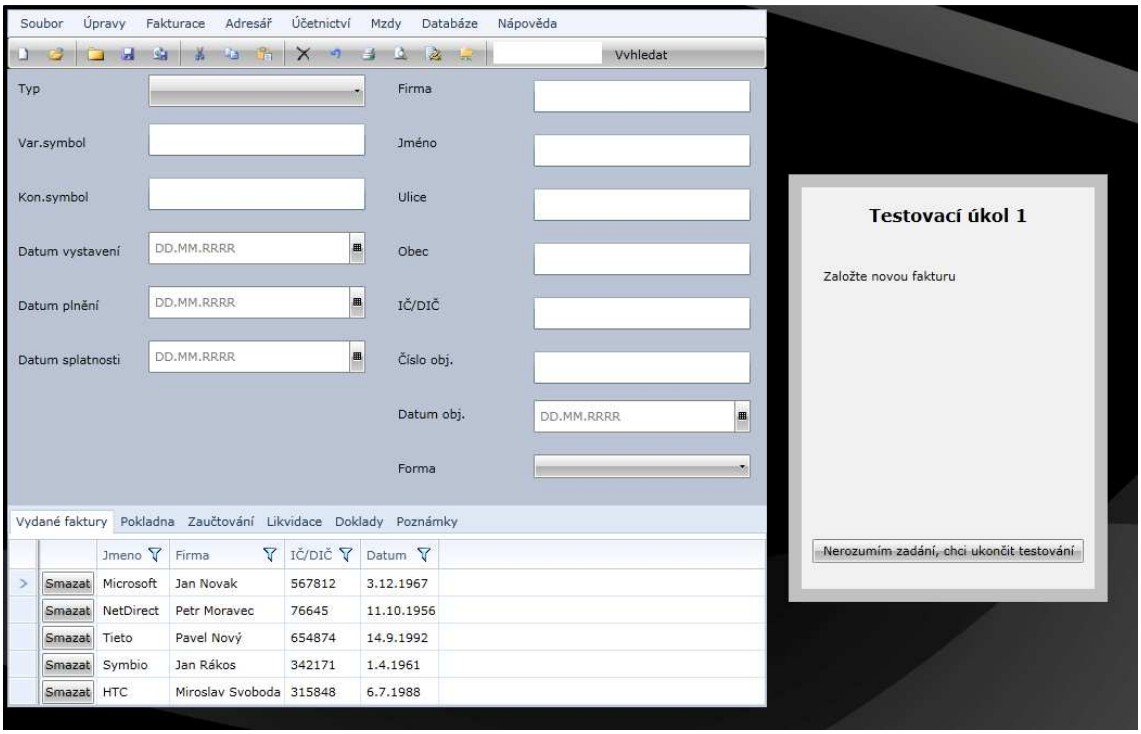
Bylo zkoumáno, zda testeři pro odeslání použijí kliku myši anebo klávesy `Enter`. Zajímavostí je, že klávesa `Enter` byla použita jen v 5% případů. Nejspíše uživatelé mají strach, že stiskem klávesy `Enter` vyvolají jinou událost, než je odeslání formuláře.

	Absolutní četnost	Relativní četnost
Myší	35	95%
Klávesa Enter	2	5%



Graf 8: Odeslání formuláře

6.3. TEST 2 – Klasické menu

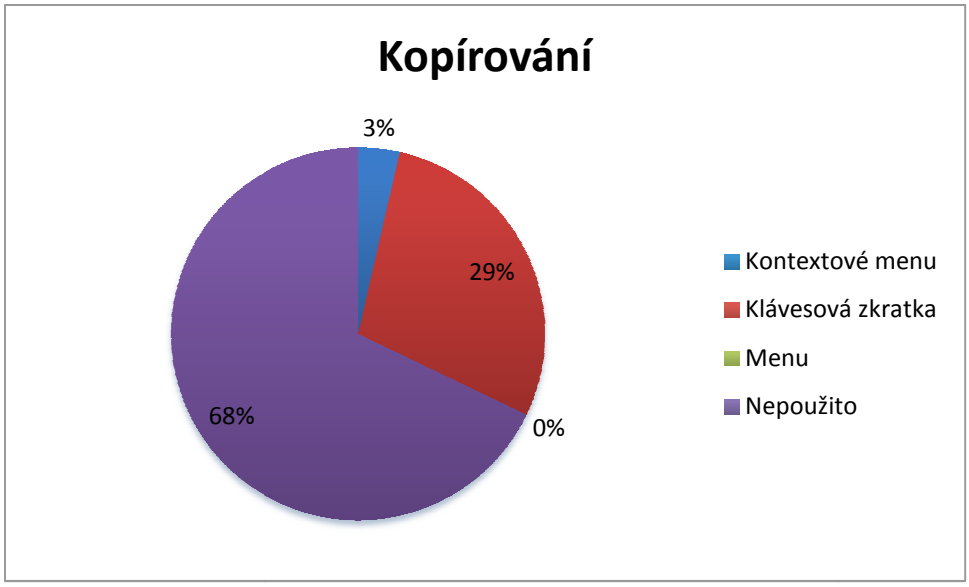


Obrázek 6-2: Test 2 - Klasické menu - ukázka obrazovky

6.3.1 Kopírování

Testeři měli do polí Konstantní symbol a Variabilní symbol zadat poměrně komplikovaná čísla. Přímo to tedy vybízelo k použití kopírování. Tuhle možnost využilo 32% testovaných, neboli 9 testovaných. Z toho 8 použilo klávesovou zkratku a jen jeden kontextovou nabídku. Kopírování pomocí menu, jak pomocí hlavního menu anebo panelu nástrojů, nepoužil nikdo.

	Absolutní četnost	Relativní četnost
Kontextové menu	1	3%
Klávesová zkratka	8	29%
Menu	0	0%
Nepoužito	19	68%

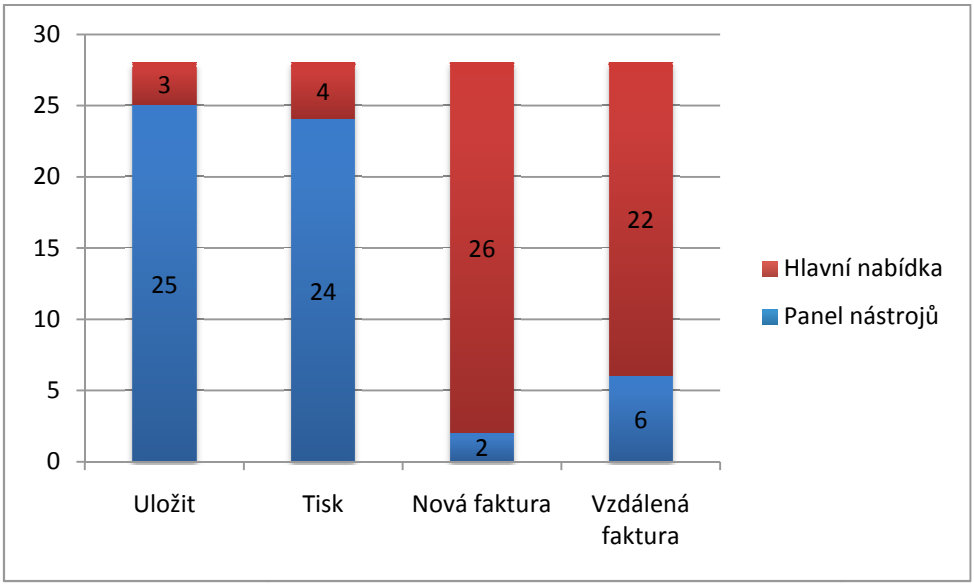


Graf 9: Kopírování

6.3.2 Spouštění příkazů

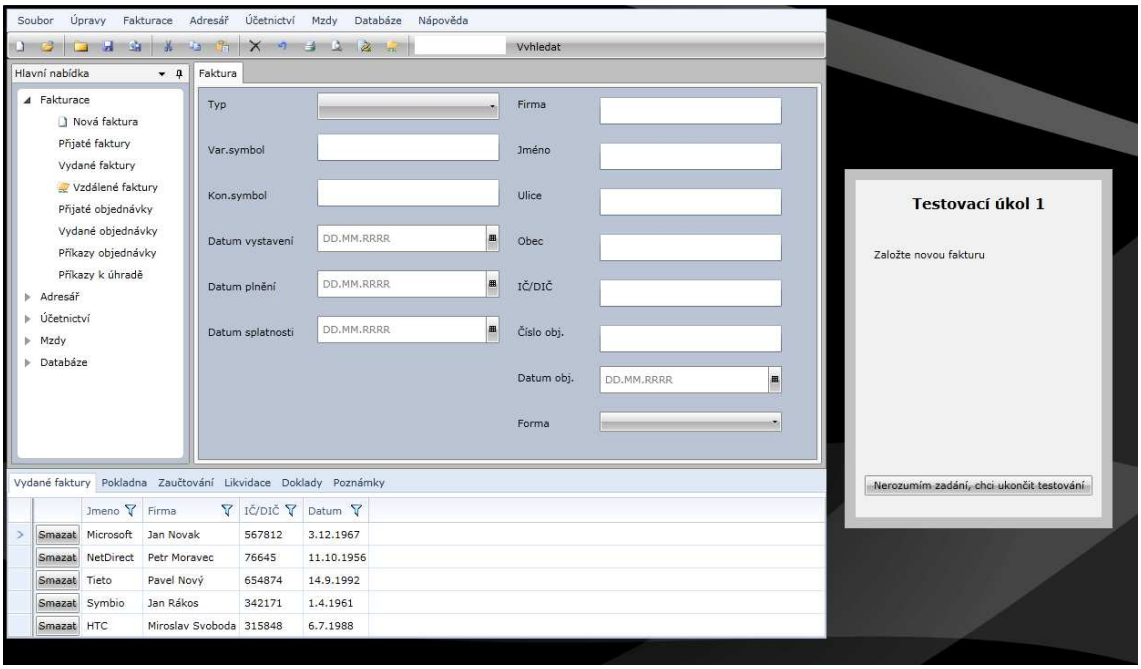
Bylo zkoumáno, zda pro spouštění funkcí dávají testeři přednost hlavní nabídce anebo ikonám v panelu nástrojů. Zkoumány byly příkazy, které se vyskytují v obou prvcích. Z grafu lze tedy vidět, že pro spouštění klasických příkazů `Uložit` a `Tisk` dávají testeři přednost nabídce nástrojů. Jde o standardní příkazy vyskytující se ve většině programů a pro jejich grafické zobrazení byly použity standardní ikony systému Windows. Naproti tomu příkazy `Nová faktura` a `Vzdálená faktura` standardní nejsou, a testeři ve většině případů dali přednost hlavní nabídce.

Byla zde zkoumána ještě jedna vlastnost, a to ta, že testeři dostanou za úkol přímo spustit příkaz z nabídky nástrojů a v následném kroku budou muset spustit příkaz, který se nachází hned vedle něho. Je tedy velká šance, že při hledání prvního příkazu na něj narazili. Ten následný příkaz byl v našem případě příkaz `Vzdálená faktura`. Jak jde vypozorovat, například ve srovnání s příkazem `Nová faktura`, že se zvýšil počet testerů, kteří pro jeho spuštění využili nabídku nástrojů. Zároveň toto zvýšení nebylo nijak výrazné, a proto se můžeme domnívat, že uživatelé pro spouštění příkazů, které nejsou standardní a v aplikaci kterou neznají, dávají přednost hlavní nabídce.



Graf 10: Využití prvků pro spouštění příkazů

6.4. TEST 3 – Postranní panel

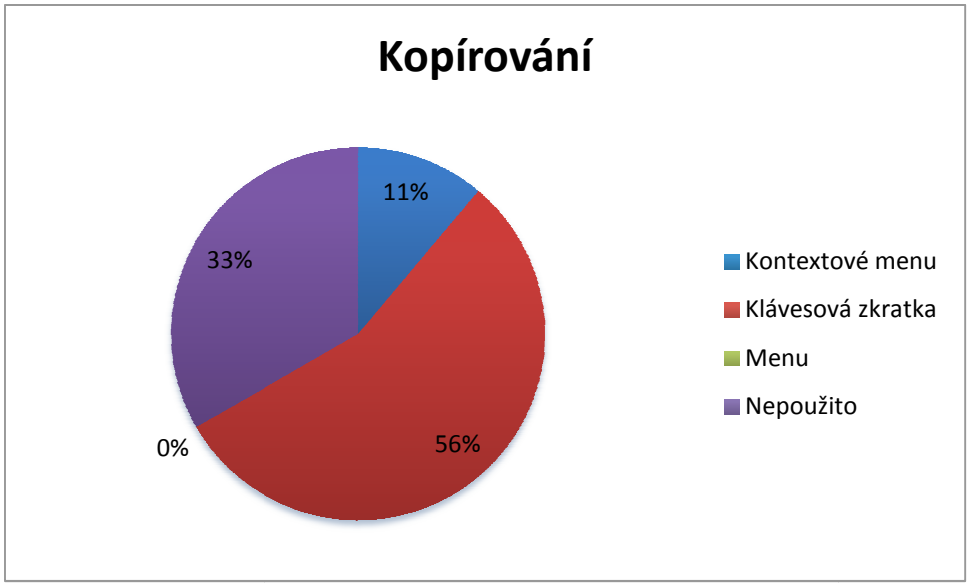


Obrázek 6-3: Test 3 - Postranní panel - ukázka obrazovky

6.4.1 Kopírování

Stejně jako v testu 2 bylo zkoumáno použití kopírování. Tuhle možnost využilo 67% testovaných, neboli 12 testovaných. Z toho 10 použilo klávesovou zkratku a dva kontextovou nabídku. Kopírování pomocí menu, jak pomocí hlavního menu anebo panelu nástrojů, nepoužil nikdo.

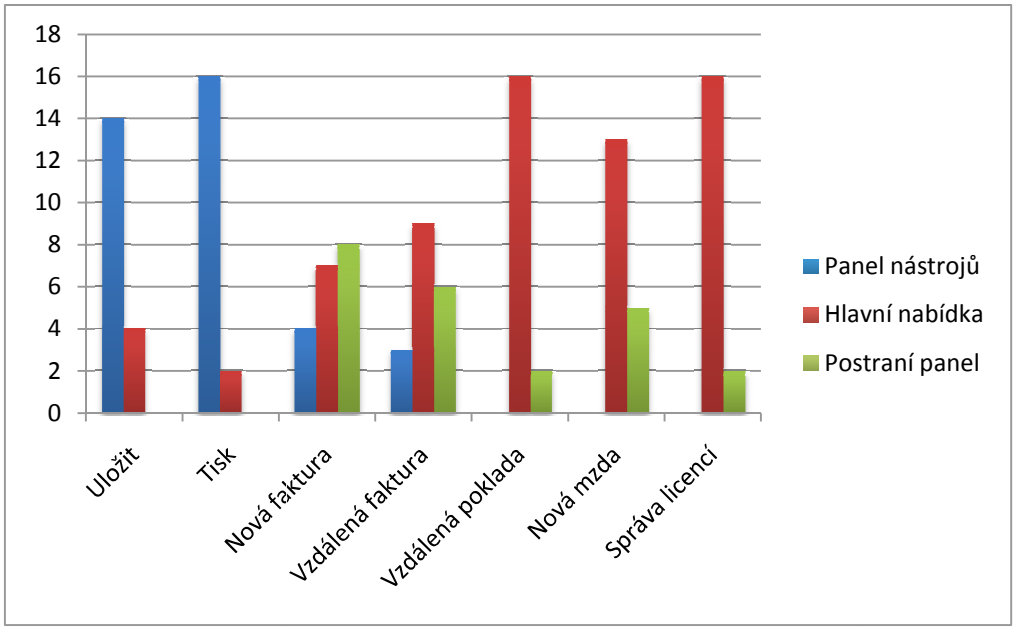
	Absolutní četnost	Relativní četnost
Kontextové menu	2	11%
Klávesová zkratka	10	56%
Menu	0	0%
Nepoužito	6	33%



Graf 11: Kopírování

6.4.2 Spouštění příkazů

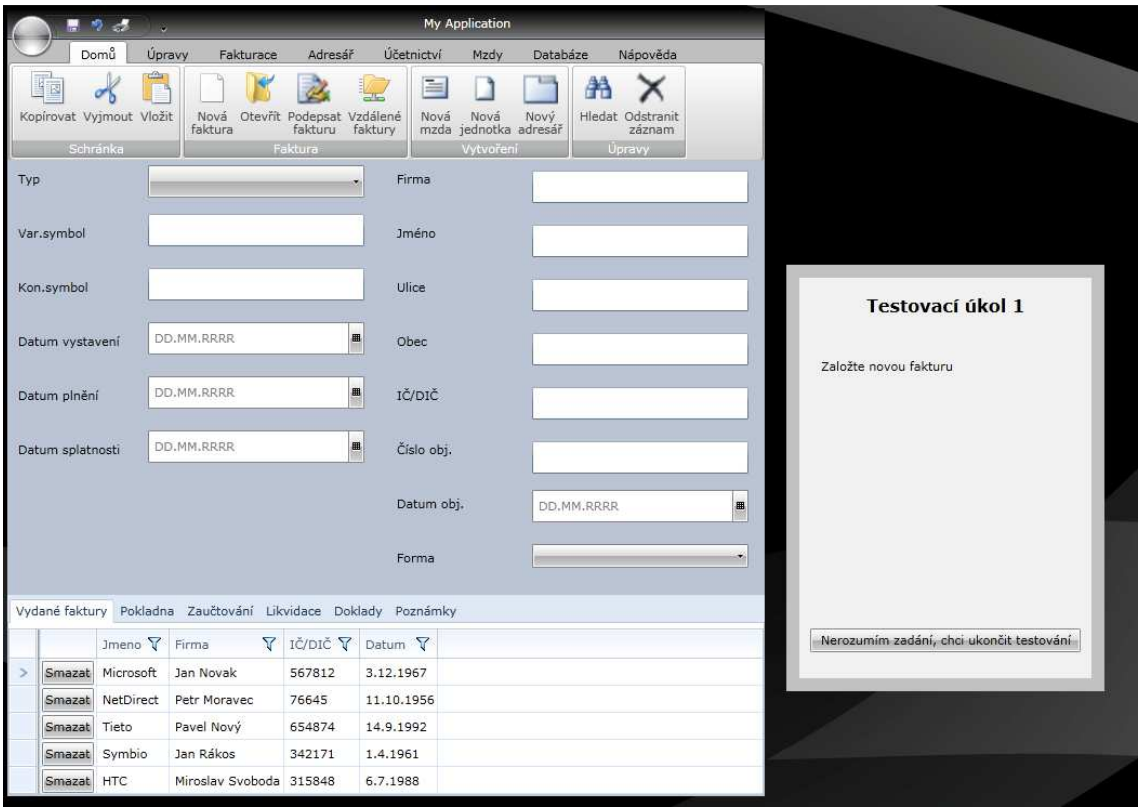
Bylo zkoumáno, zda pro spouštění funkcí dávají testeři přednost hlavní nabídce, ikonám v panelu nástrojů anebo postrannímu panelu. Dalo by se říci, že příkazy `Uložit` a `Tisk` se vyskytovaly pouze v panelu nástrojů a hlavní nabídce a naopak `Vzdálená pokladna`, `Nová mzda` a `Správa licencí` jen v hlavní nabídce a postranním panelu. Z grafu lze tedy vidět, že pro spouštění klasických příkazů `Uložit` a `Tisk` dávají testeři přednost panelu nástrojů. Jde o standardní příkazy vyskytující se ve většině programů a pro jejich grafické zobrazení byly použity standardní ikony systému Windows. Naproti tomu příkazy `Nová faktura` a `Vzdálená faktura` standardní nejsou. Proto v případě `Nové faktury` dali testeři v nejvíce případech přednost postrannímu panelu. U `Vzdálené faktury` zase dali přednost hlavní nabídce. Svou roli nejspíše sehrálo to, že v postranním panelu byla u `Nové faktury` ikona a u vzdálené nikoliv. V ostatních případech, kdy se příkazy vyskytovaly jen v postranním panelu a hlavní nabídce, vítězí vždy hlavní nabídka.



Graf 12: Využití prvků pro spouštění příkazů

Při zkoumání postupu, kdy testeři dostanou za úkol přímo spustit příkaz z panelu nástrojů a v následném kroku budou muset spustit příkaz `Vzdálená faktura`, který se nachází hned vedle něho, bylo zjištěno, že použití panelu nástrojů pro `Vzdálenou fakturu` bylo ještě menší než pro založení `Nové faktury`.

6.5. TEST 4 – Ribbon

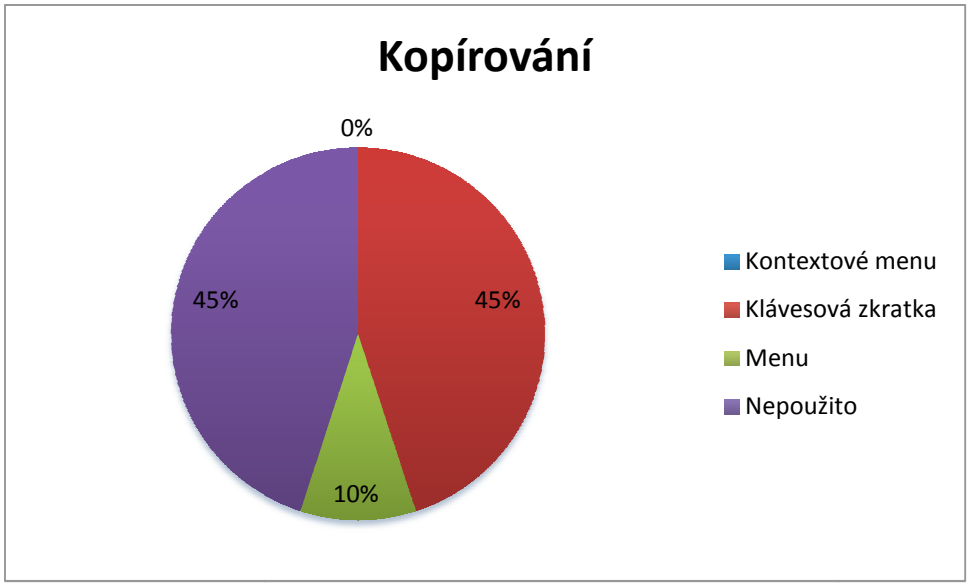


Obrázek 6-4: Test 4 - Ribbon - ukázka obrazovky

6.5.1 Kopírování

Stejně jako v testu 2 a 3 bylo zkoumáno použití kopírování. Tuhle možnost využilo 55% testovaných, neboli 11 testovaných. Z toho 9 použilo klávesovou zkratku a pomocí menu, jak pomocí hlavního menu anebo panelu nástrojů, pouze 2 testeři. Kontextové menu nepoužil nikdo.

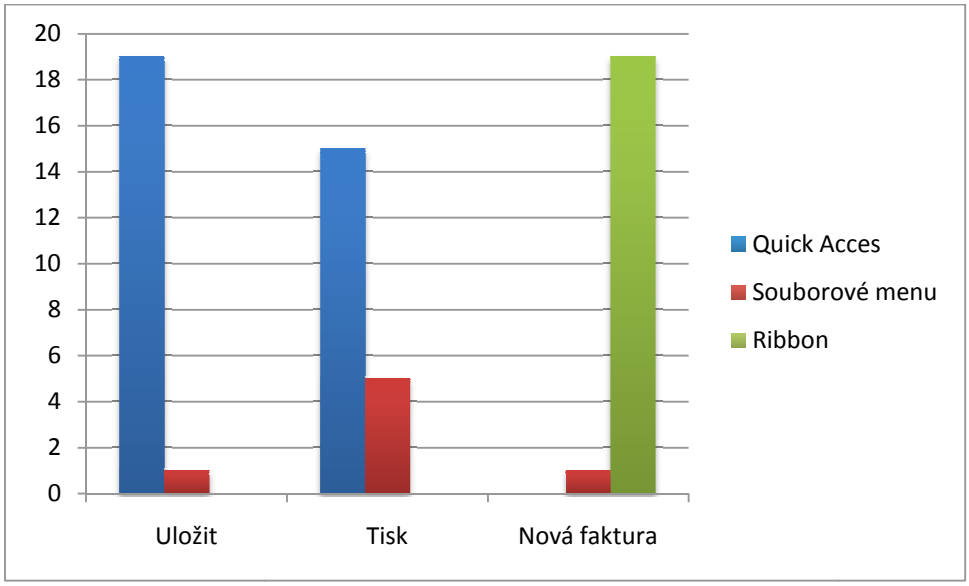
	Absolutní četnost	Relativní četnost
Kontextové menu	0	0%
Klávesová zkratka	9	45%
Menu	2	10%
Nepoužito	9	45%



Graf 13: Kopírování

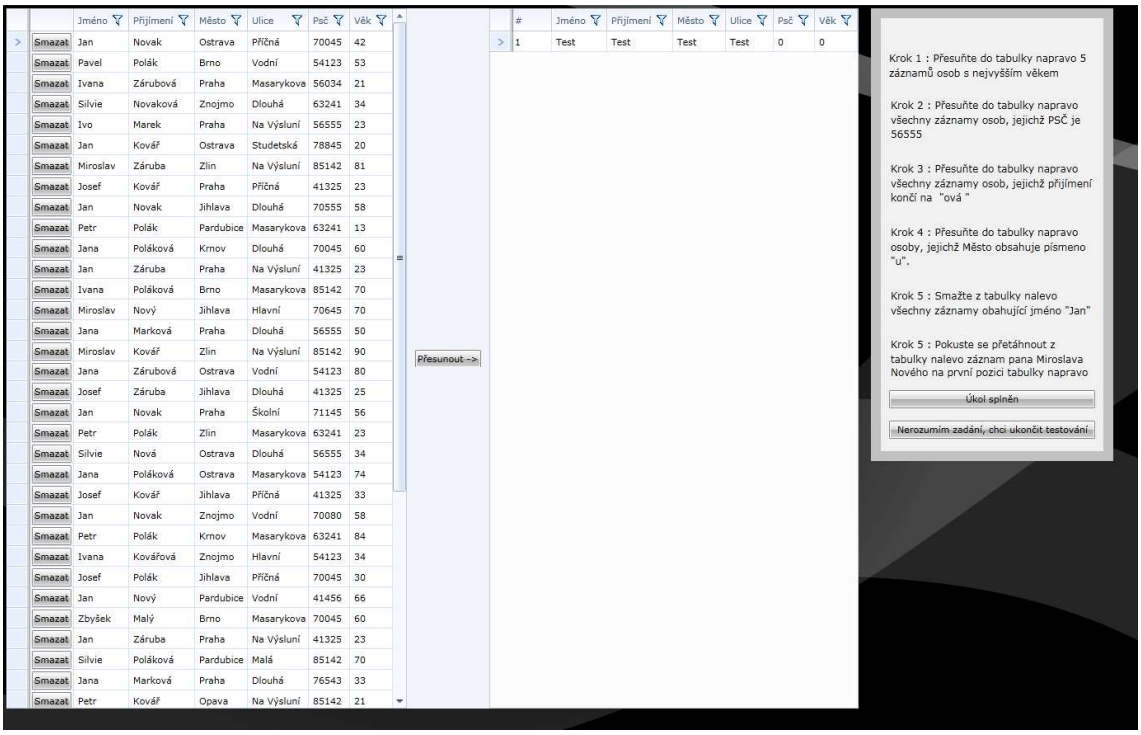
6.5.2 Spouštění příkazů

V tomhle případě bylo zkoumáno, zda pro spouštění funkcí dávají testeři přednost Quick Acces, souborovému menu anebo čistě jen Ribbonu. Dá se říci, že příkazy `Uložit` a `Tisk` se vyskytovaly pouze v Quick Acces a souborovém menu a naopak `Nová faktura` jen v souborovém menu anebo Ribbonu. Z grafu lze tedy vidět, že pro spouštění klasických příkazů `Uložit` a `Tisk` dávají testeři přednost Quick Acces panelu. Jde o panel poměrně nový, ale je vidět, že uživatelé jsou na něj zvyklí a dávají mu přednost. Při příkazu `Nová faktura` dali testeři ve většině případů přednost Ribbonu před souborovým menu. V ostatních případech se příkazy vyskytovaly už jen v Ribbonu.



Graf 14: Využití prvků pro spouštění příkazů

6.6. TEST 5 - Práce s tabulkami

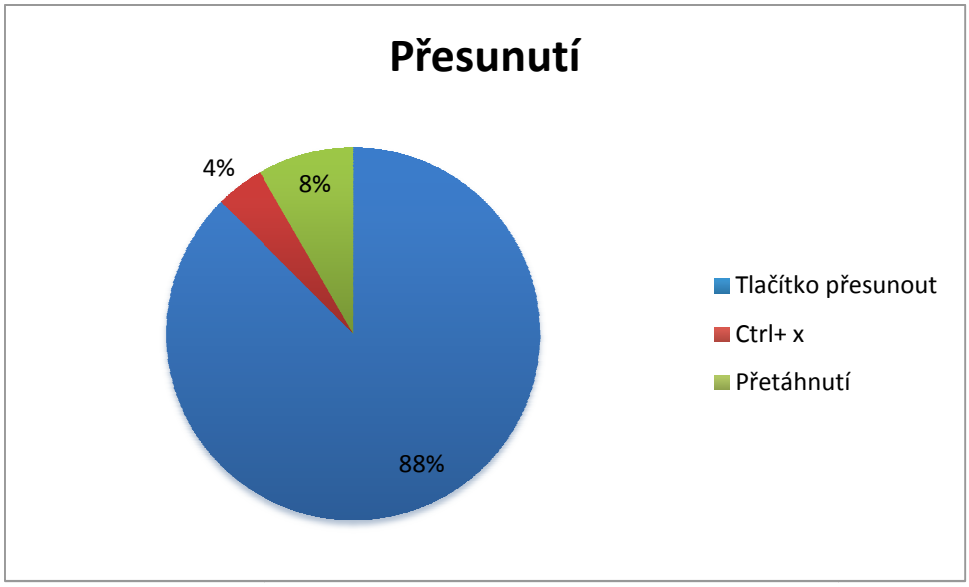


Obrázek 6-5: Test 5 - Práce s tabulkou - ukázka obrazovky

6.6.1 Přesunutí prvků

V tomto případě bylo zkoumáno, zda testeři pro přesunutí buněk z jedné tabulky do druhé použili tlačítko `Přesunout`, klávesovou zkratku `Ctrl+x` anebo přetáhnutí (Drag and Drop). V nejvíce případech použili tlačítko `Přesunout`. `Ctrl+x` a přetáhnutí použilo jen malé procento testerů. Přetáhnutí měli díky poslednímu úkolu použít všichni uživatelé minimálně jednou, a to taky až na jeden případ všichni splnili. Pouze dva testeři však využili přetáhnutí vícekrát než jednou. Jelikož dokončených testů bylo 22, někteří testeři použili více způsobů.

	Absolutní četnost	Relativní četnost
Tlačítko přesunout	21	88%
Ctrl + x	1	4%
Přetáhnutí	2	8%

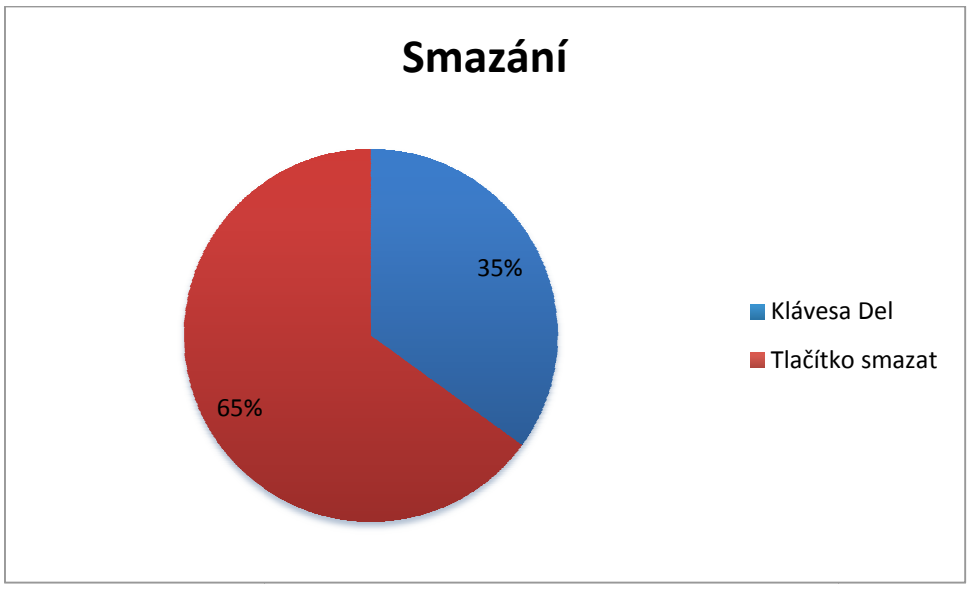


Graf 15: Přesunutí

6.6.2 Smazání

Zde bylo zkoumáno, zda pro smazání buňky testeři použili tlačítko `Smazat` anebo klávesu `Delete`. V 65% použili tlačítko a to i přesto, že často bylo nutno smazat více buněk zároveň. Jelikož dokončených testů bylo 22, někteří tento úkol vůbec nesplnili.

	Absolutní četnost	Relativní četnost
Klávesa Del	7	35%
Tlačítko smazat	13	65%

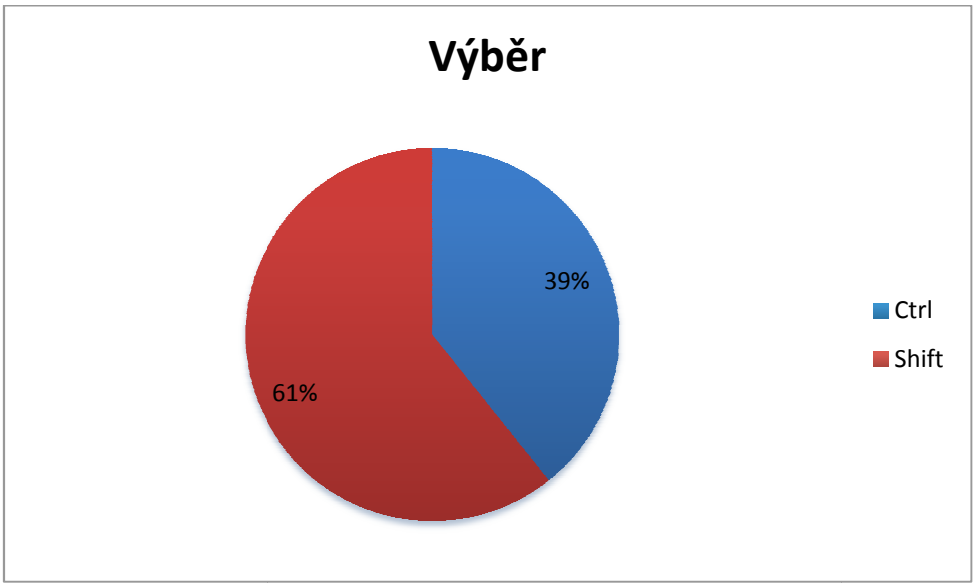


Graf 16: Smazání

6.6.3 Výběr

Při práci s tabulkami je často potřeba vybírat více buněk zároveň. Pro tzv. „multiselect“, výběr více buněk zároveň, často slouží klávesy `Ctrl` anebo `Shift`. Zaznamenáno vždy bylo, zda daný uživatel zmáčkl klávesu `Shift` a zda zmáčkl klávesu `Ctrl`. Jelikož dokončených testů bylo 22, někteří použili oba způsoby. Celkově však v našem případě dalo více uživatelů přednost klávese `Shift`.

	Absolutní četnost	Relativní četnost
Ctrl	11	39%
Shift	17	61%

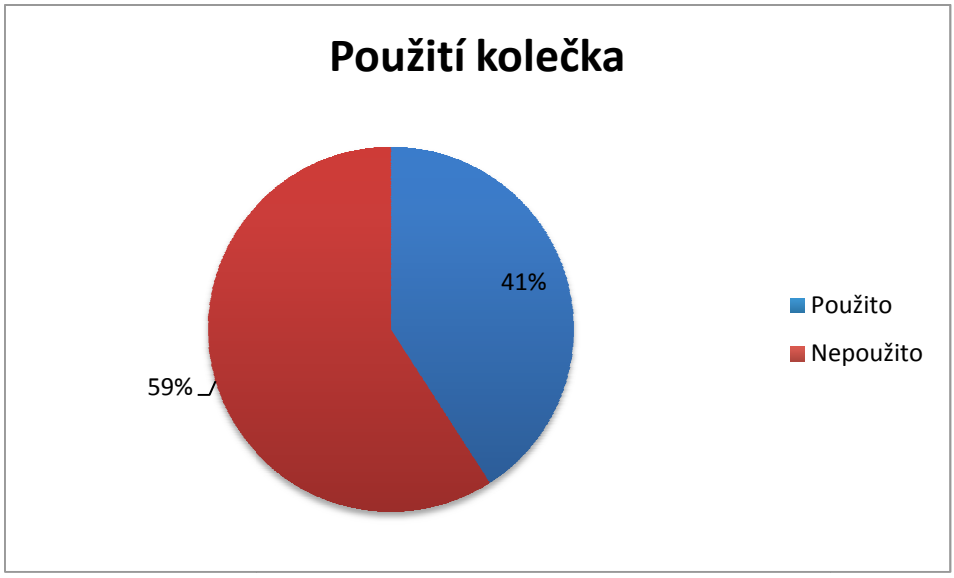


Graf 17: Výběr

6.6.4 Použití kolečka

Jelikož byly tabulky v Testu 5 poměrně rozměrné, uživatelům s menším rozlišením se nevešly do okna a ti tak museli použít skrolování. Zároveň i počet záznamů v tabulkách byl větší, než byla velikost tabulek, takže i v tomto případě museli skrolovat. Výsledkem bylo, že 41% uživatelů použilo pro skrolování kolečko myši, což je poměrně více jak 19% v Testu 1, ale stále je to nízké číslo. Pro pohyb mezi buňkami bylo požito kláves `nahoru` a `dolů` jen ve 3 případech.

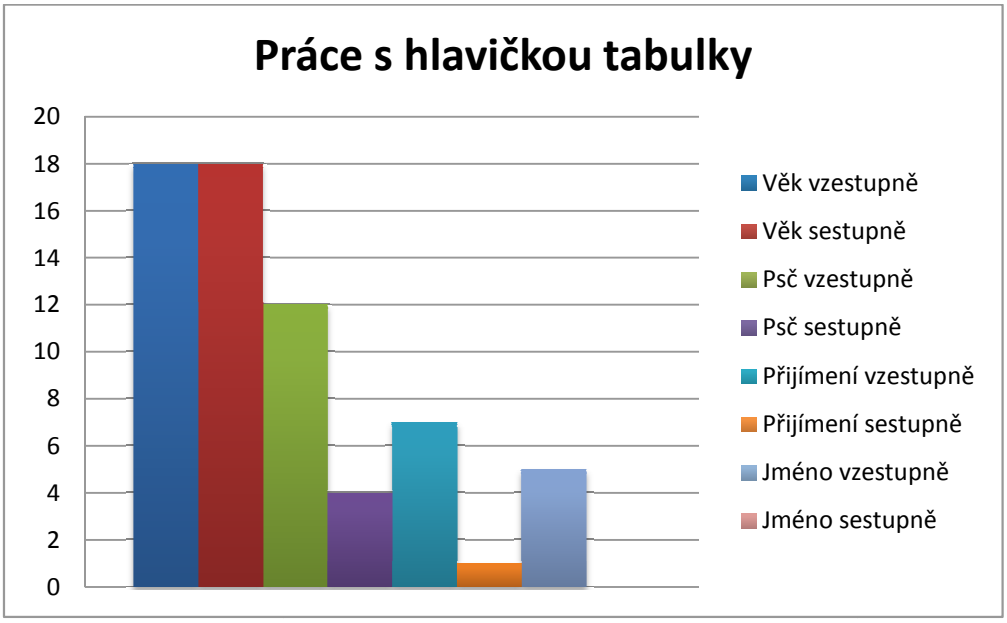
	Absolutní četnost	Relativní četnost
Použito	9	41%
Nepoužito	13	59%



Graf 18: Použití kolečka

6.6.5 Práce s hlavičkou tabulky

Celkem běžné chování u tabulek je to, že po kliknutí na záhlaví daného sloupce se prvky seřadí podle dané hodnoty vzestupně anebo sestupně. V našem případě danou funkčnost využilo 91% všech testerů. Jde vidět, že tato funkce je jim dobře známa. Například běžné chování je, že při prvním kliknutí na záhlaví se buňky seřadí vzestupně, při dalším sestupně a následně se seřazení zruší. V první úkolu například měli testeři vybrat 5 záznamů osob s nejvyšším věkem. Proto klikli na záhlaví věk jednou a poté znovu pro sestupné seřazení. Je taky vidět, že s postupem dalších úkolů klesalo množství seřazení, nejspíše díky tomu, že klesal i počet záznamů, a tak uživatelé dávali přednost přímému výběru.



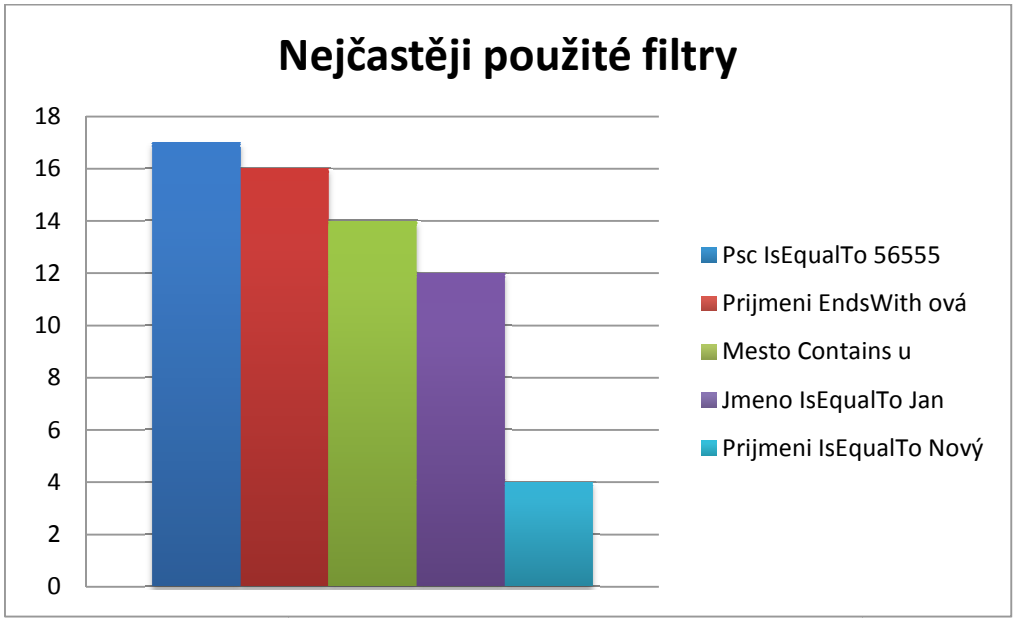
Graf 19: Práce s hlavičkou tabulky

6.6.6 Práce s filtry

V našem testovacím případě bylo možno pro složitější výběry použít i filtry. Ty byly přístupné po kliknutí na ikonku „trychtýře“ v hlavičce každého sloupce. Nejedná se přímo o standardní záležitost, ale i přesto ji v našem případě využilo 86% všech testerů.

Úkoly 2-6 byly přímo koncipovány tak, aby využily funkce filtrů jako IsEqualTo (rovná se), EndsWidth (končí na), Contains(obsahuje) atd.

Z grafu 20 lze vidět, že stejně jako v případě seřazení, tak i zde s postupem dalších úkolů klesalo množství použitých filtrů, nejspíše díky tomu, že klesal i počet záznamů, a tak uživatelé dávali přednost přímému výběru.



Graf 20: Nejčastěji používané filtry

Pro připomenutí, pro implementaci filtrů byla použita komponenta od firmy Telerik. Na práci s ní si však testeři stěžovali. Výtky byly k tomu, že ikona, signalizující že je filtr aktivní, je špatně vidět. Bylo by lepší, kdyby se okno s filtry zavíralo křížkem, a kdyby bylo nějaké centrální tlačítko pro vymazání všech filtrů. Při kontrole použitých filtrů se taky zjistilo, že 9 uživatelů, to je 41% všech testovaných, nepoužívalo tlačítko `Smazat filtr`. Vždy, když chtěli filtr zrušit, a zobrazit zpět všechny prvky, udělali to tak, že vybrali všechny prvky pomocí tlačítka `Select All` a dali filtrovat. Z toho se lze domnívat, že filtrování buněk v podání Teleriku není zrovna uživatelsky přívětivé, a to i s přihlédnutím k tomu, že testeři jsou vesměs zkušení, až velmi zkušení uživatelé.

6.6.7 Srovnání použití seřazení a filtrů

Pokud bychom měli srovnat seřazení pomocí kliknutí na hlavičku tabulky a filtry, tak 91% všech testerů použilo seřazení pomocí kliknutí na hlavičku tabulky a 86% filtrů. Jsou to

poměrně vysoká procenta, ale opět musíme přihlédnout ke schopnostem testovaných. Zkoumána byla i možnost, zda použití těchto prvků závisí na uvedených zkušenostech, ale tato domněnka nebyla prokázána. Zároveň lze vidět, že filtry a seřazení využilo přibližně stejné množství uživatelů.

To ale neříká nic o tom, zda pro řešení úkolů dávali testeři přednost filtrům anebo seřazení pomocí kliknutí na hlavičku tabulky. Např. úkol 2, který měl zadání: `Přesuňte do tabulky napravo všechny záznamy osob, jejichž PSČ je 56555`, se dá vyřešit 3 způsoby, a to použitím filtru Psč isEqualTo 56555, seřazením sloupce Psč a vybráním buňek obsahujících 56555 a nebo ručně, výběrem buněk s Psč 56555. V našem případě 17 testerů použilo Psč isEqualTo 56555, ale zároveň 12 testerů použilo seřazení Psč. Zajímavostí je, že testerů bylo 23 a tak tedy některý musel využít obě varianty, neboli zvolit jednu, a poté druhou. V úkolu 2 a poté i ve všech ostatních úkolech, až na úkol poslední, dali testeři větší přednost filtrům před seřazením.

Lze taky vysledovat, že se stoupajícím počtem úkolů klesal počet seřazení a filtrů, a uživatelé dávali přednost přímému výběru. Nejspíše proto, že počet záznamů byl menší, a tak byl pro ně snažší přímý výběr než použití filtrů a nebo seřazení. Proto například na posledním úkolu, kdy měli vybrat specifický záznam, bylo použito jen 4x filtru a 7x seřazení. Takže při posledním úkolu, kdy z původního počtu záznamů 46 zůstalo už jen třináct, už dala více jak polovina testovaných přednost přímému vyhledání před použitím filtrů a nebo seřazení.

6.6.8 Použití filtrů a seřazení a závislost na čase

V prvním případě mělo být ověřeno, zda počet seřazení ovlivňuje čas vykonání testů. Jednotlivé testy byly tedy rozděleny do skupin podle počtu seřazení a jejich časy poté byly porovnány pomocí neparametrického Kruskal-Wallisova testu [32]. Tento test byl vybrán, jelikož nevyžaduje normalitu rozdělení dat, kterou daný výběr nesplňuje. Data však musí splňovat podmínku homoskedasticity, identických rozptylů.

Nejdříve byla ověřena podmínka homoskedasticity. Ta byla potvrzena, avšak ne ve všech testech. Poté se pokračovalo v Kruskal-Wallisově testu.

Jako nulová hypotéza byla zvolena možnost, že počet seřazení čas vykonání testů neovlivňuje. P-Value vyšlo, že se rovná 0,5825, což je více jak 0,05. Takže nebyla zamítnuta nulová hypotéza. Dá se tedy říct se 95% spolehlivostí, že počet použitých seřazení nemá významný vliv na celkový čas vykonání testu.

Stejný test byl poté proveden i pro ověření, zda počet použitých filtrů ovlivňuje čas vykonání testů. Jednotlivé testy byly rozděleny do skupin podle počtu filtrů. Problém však byl u těch případů, kdy uživatelé nepoužívali zrušení filtru, ale označili všechny hodnoty. Takové testy poté obsahovaly počet filtrů v řádu desítek, proto byly umístěny do speciální skupiny.

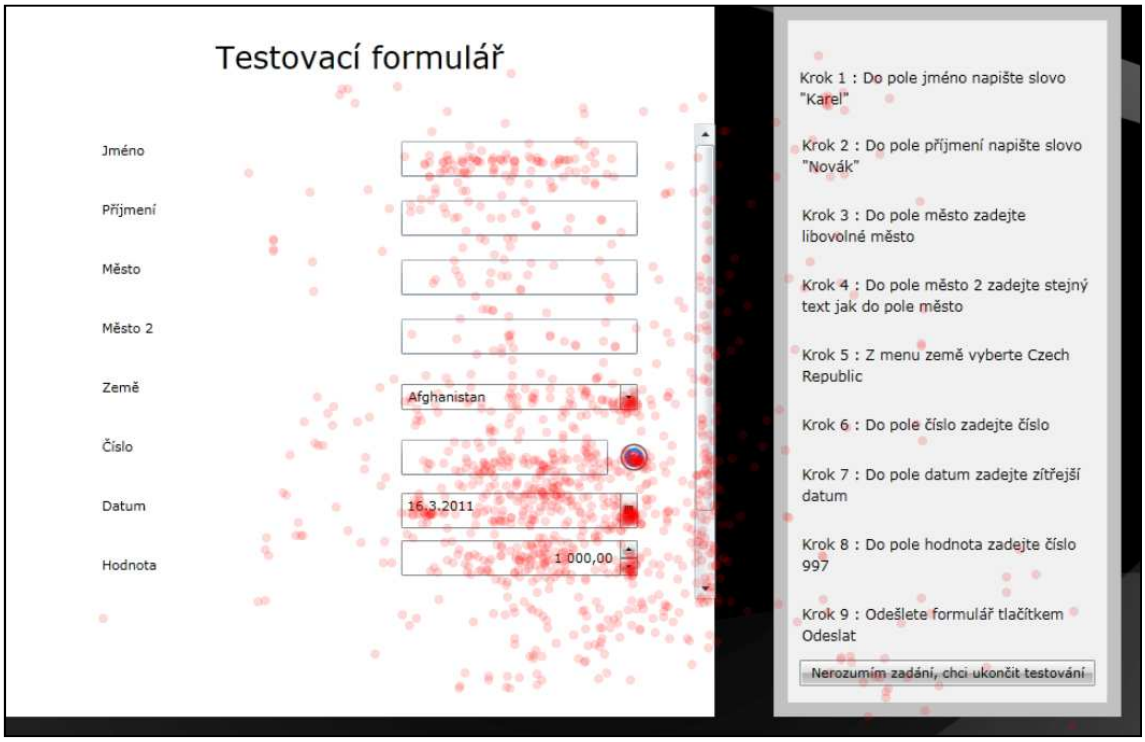
Nejdříve byla ověřena podmínka homoskedasticity. Ta byla potvrzena, a tak se pokračovalo v Kruskal-Wallisově testu.

Jako nulová hypotéza byla zvolena možnost, že počet použitých filtrů čas vykonání testů neovlivňuje. P-Value vyšlo, že se rovná 0,7556, což je více jak 0,05. Takže nebyla zamítnuta nulová hypotéza. Dá se tedy říct se 95% spolehlivostí, že počet použitých filtrů nemá významný vliv na celkový čas vykonání testu.

6.7. Vyhodnocení pohybů myši

Do stávající Silverlight aplikace pro testování byla přidána funkcionalita pro vyhodnocení výsledku pohybu myši. U každého testu byla totiž snímána každých 0,5 sekund poloha myši a ukládána do databáze. Byla tedy vytvořena funkce, která dané hodnoty od všech uživatelů z databáze zpracuje a zobrazí je ve formě částečně průhledných červených teček. Problémem však bylo, že aplikace byla vytvořena tak, aby testy kvůli vzhledu byly vždy umístěné na střed okna. Jenže díky různým rozlišením zobrazovacích zařízení testerů by byly hodnoty zkreslené. Proto musel být vytvořen algoritmus, který zachycené souřadnice myši přetransformuje podle daného rozlišení. Pro přehlednost ještě bylo nastaveno, že se nebudou započítávat stejné hodnoty za sebou. Takže pokud např. tester 10 sekund stál s myší na jednom místě, tak se na dané pozici zobrazí ve výsledku jen jedna tečka.

6.7.1 Test 1 – Formulář

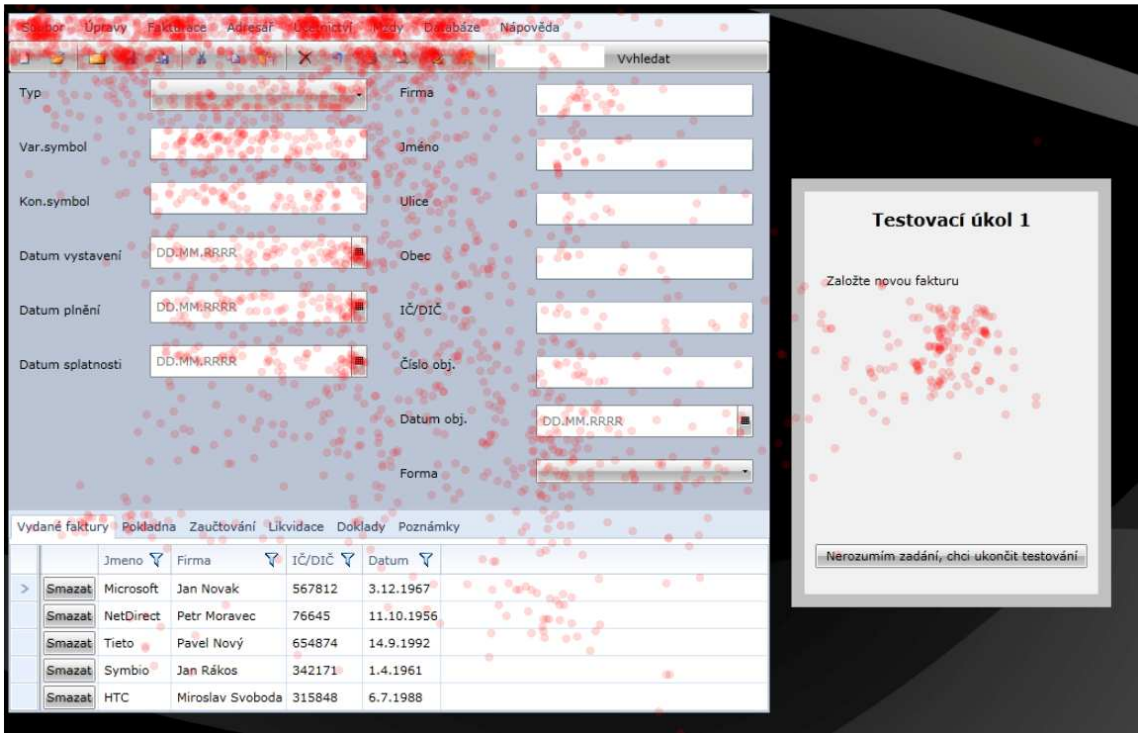


Obrázek 6-6: Test 1 - Formulář - vyhodnocení pohybu myši

U testu 1 je například hezky vidět, že v prvním poli Jméno se myš vyskytovala mnohem více než v ostatních polích. To by odpovídalo chování, kdy uživatel klikne do prvního pole a dále pro přechod mezi poli používá již jen klávesu `TAB`. Odpovídalo by to i výsledkům grafu 23, kdy je jasně vidět, že uživatelé pro přechod mezi poli dávají přednost klávese `TAB` v 89% případů. Je taky vidět, že po výběru země z rozbalovacího menu poté opět klikli do políčka číslo, a

nepoužili klávesu TAB. A to samé u data, kde lze vidět, že část testovaných dala přednost výběru pomocí kalendáře.

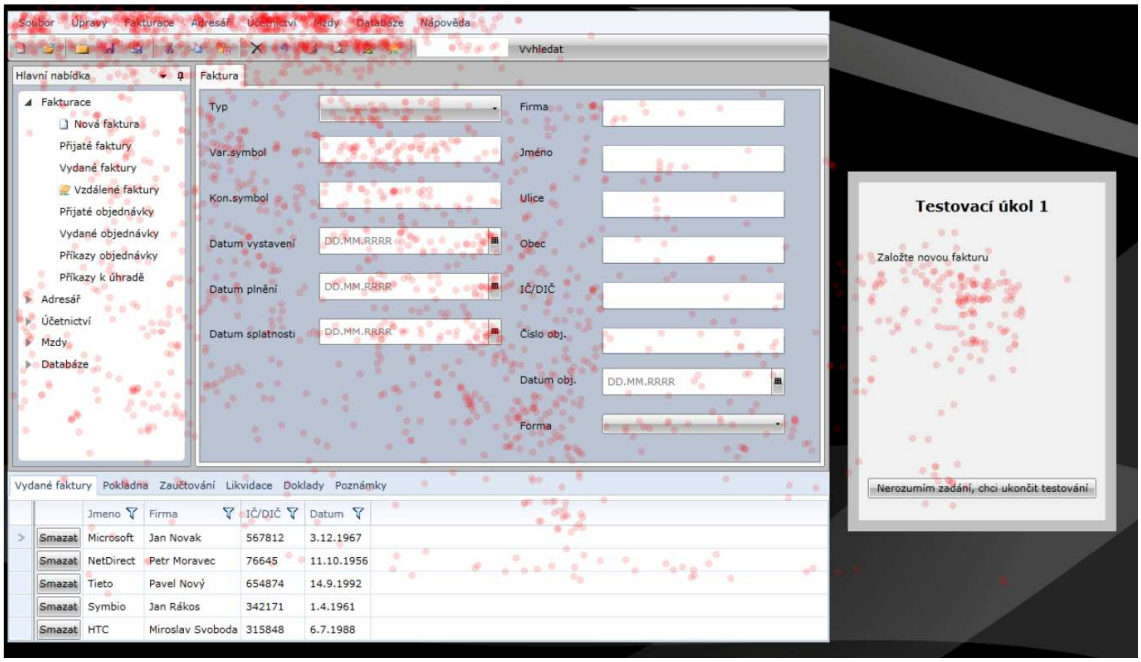
6.7.2 Test 2 – Klasické menu



Obrázek 6-7: Test 2 - Klasické menu - vyhodnocení pohybu myši

Zde lze vypozorovat, že většina pohybů myší se konala v oblasti hlavní nabídky a panelu nástrojů. Zajímavé je, že testeři myší pohybovali i na ikony, které v testu neměly žádný význam. Lze tedy hezky vidět nešvar daného rozhraní, že když uživatel potřebuje zjistit, jakou funkci zastupuje daná ikona, musí na ni umístit myš a vyčkat na zobrazení názvu. Za zmínku stojí i to, že na rozdíl od prvního testu, kdy testeři klikli na první pole a poté procházeli prvky pomocí klávesy TAB, v tomhle případě je vidět, že poměr myši na jednotlivých polích je přibližně stejný. Odpovídalo by to i výsledkům z grafu 23, kdy uživatelé pro přechod mezi poli dávají přednost klávese TAB už jen v 57% případů. Vychází tedy, že tento způsob ovládání je více zažitý při práci s formuláři než aplikacemi.

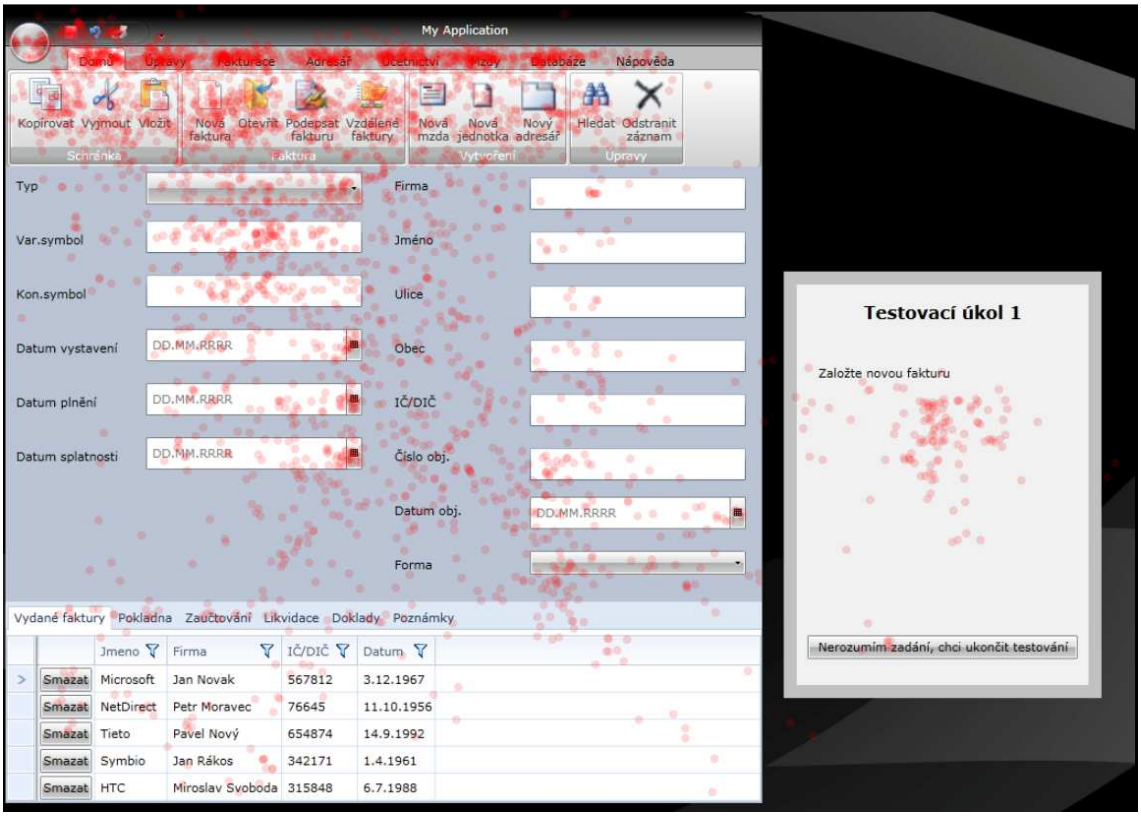
6.7.3 Test 3 – Postranní panel



Obrázek 6-8: Test 3 - Postranní panel - vyhodnocení pohybu myši

Závěry z Testu 2 by se daly vztáhnout i na Test 3, ve věcech týkajících se procházení polí a práci s panelem nástrojů. Dále je zde vidět, že většina testerů dala přednost procházení hlavního menu před postranním panelem.

6.7.4 Test 4 – Ribbon



Obrázek 6-9: Test 4 - Ribbon - vyhodnocení pohybu myši

U Testu 4 je vidět, že většina pohybu myši byla vykonávána v oblasti oušek přepínacích karet rozhraní Ribbon. Zajímavý však je velký počet teček i v oblasti ikon. Přitom na rozdíl od rozhraní Testu 2, kdy se na ikonu musí najet myší, aby se ukázal popis, není toto v tomhle případě potřeba. Textový popis u ikon je zobrazen, a tak není nutnost na ně najíždět. Z analýzy jde taky vypozorovat celkem výrazné používání Quick Acces panelu, ikony `Uložit` a `Tisk`. U procházení polí se dá říct to samé, jako v případě Testu 2 a 3.

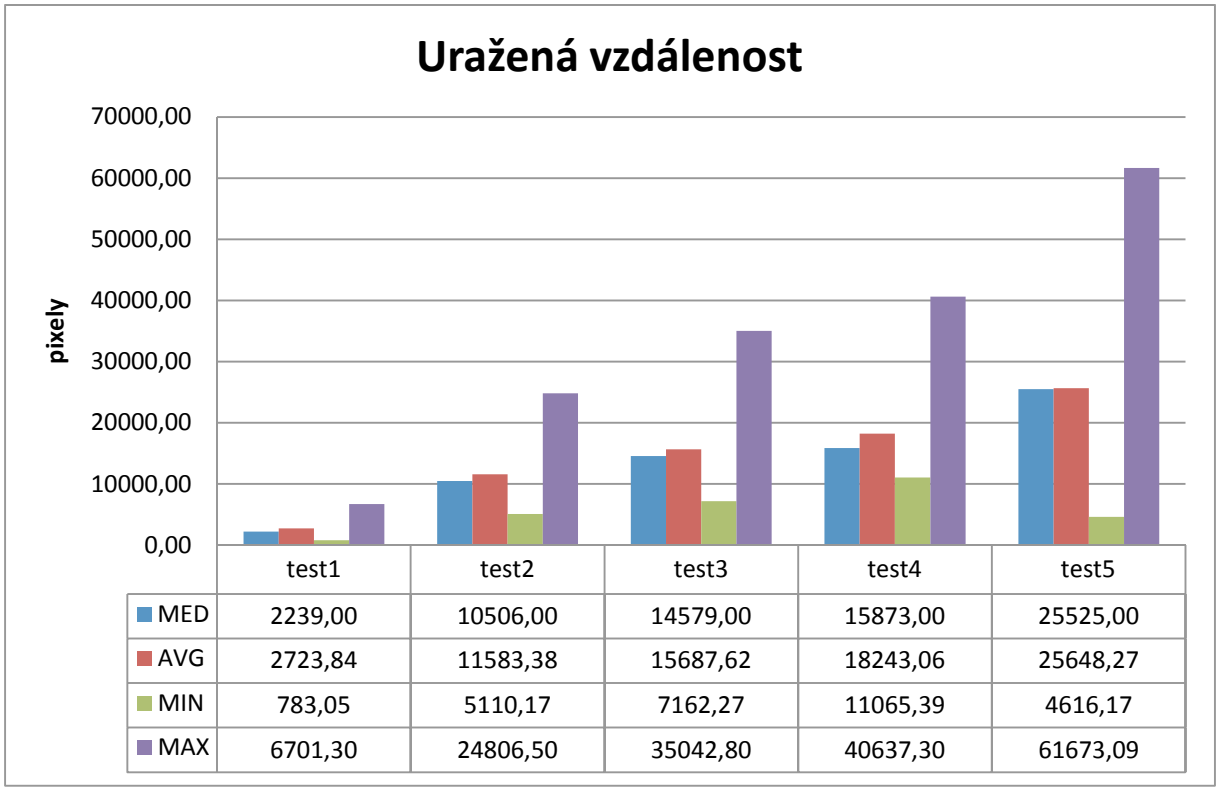
6.7.5 Test 5 – Práce s tabulkami

Obrázek 6-10: Test 5 - Práce s tabulkou - vyhodnocení pohybu myši

U Testu 5 se dá hezky vypozorovat používání filtrů, seřazení, tlačítka `Přesunout` a taky tlačítek `Smazat`. Zajímavé je použití postranního prostoru vedle tabulky vlevo. Použitá komponenta tabulka má totiž tu vlastnost, že po dvojkliku na položku tabulky se zobrazí možnost editování záznamu. Pro zrušení edice se potom v postranním panelu zobrazí tlačítko. Je vidět, že i když jsme po uživatelích editaci nechtěli, několik z nich ji buď úmyslně anebo nevědomky spustilo.

6.8. Vyhodnocení uražené vzdálenosti

Testovací aplikace snímala pohyb myši a počítala uraženou vzdálenost v pixelech. Vzdálenost byla počítána jako rozdíl staré a nové hodnoty každých 0,5 sekund. V grafu 21 jsou poté uvedeny mediány těchto hodnot, průměry, minima a maxima.



	test1	test2	test3	test4	test5
MED	2239,00	10506,00	14579,00	15873,00	25525,00
AVG	2723,84	11583,38	15687,62	18243,06	25648,27
MIN	783,05	5110,17	7162,27	11065,39	4616,17
MAX	6701,30	24806,50	35042,80	40637,30	61673,09

Graf 21: Uražená vzdálenost

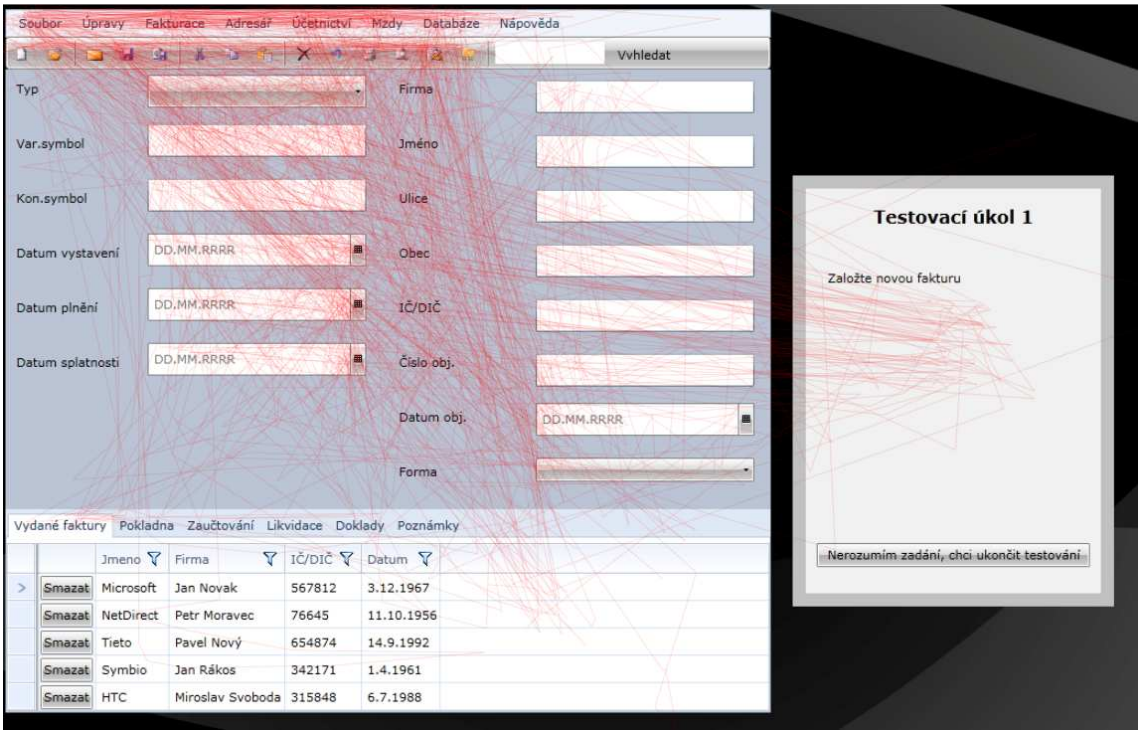
6.8.1 Test 1 - Formulář

Při analýze pohybu při testu 1 se ukázalo, že průměrná uražená vzdálenost je kolem 2700 px. Je to hodnota poměrně malá. Nejspíše je to dáno tím, že prvky byly blízko u sebe, a velké procento testerů pro přechod mezi poli použilo klávesu `TAB`.

6.8.2 Testy 2-4

Při pohledu na Graf 21: Uražená vzdálenosti lze jasně vidět, že nejmenší vzdálenost urazili testeři při testu 2. O něco větší vzdálenost urazili testeři při testu 3 a největší vzdálenost byla uražena u testu 4. Pro objasnění, proč byl některý z testů pomalejší, byly vytvořeny 3 obrazovky, které ukazují pohyb myší. Vesměs šlo o propojení všech nasbíraných bodů v pořadí, v jakém byly zaznamenány. Samotné obrazovky sice nejsou díky zobrazení všech bodů natolik názorné, ale i tak z nich lze mnohé vyčíst.

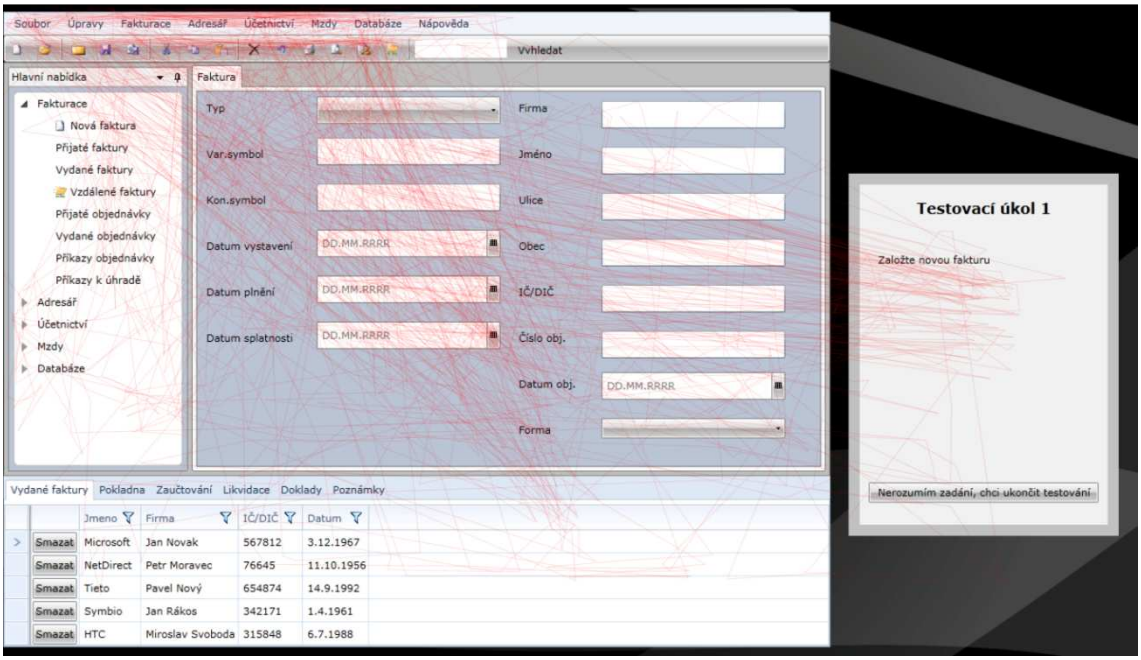
6.8.2.1. Test2 – Klasické menu



Obrázek 6-11: Test 2 - Klasické menu - vyhodnocení pohybu myši pomocí čar

Z obrázku 6-11 a ze závěrů z testů pohybů myši je vidět, že se uživatelé pohybovali vesměs jen horizontálně po hlavní nabídce a menu. Pár čar poté ukazuje pohyb a vyplňování polí, několik z nich zachycuje, jak byly kopírovány hodnoty z pravého panelu se zadáním. Čáry směřující na střed, vyskytující se u všech obrazovek, ukazují pohyb myší vykonaný při odkliknutí upozornění při splnění úkolu. Poloha upozornění byla umístěna na střed stránky, na rozdíl od prvků rozhraní, a proto se jeho vertikální poloha měnila v závislosti na rozlišení. To je důvod toho, že čáry nevedou vždy na přesnou pozici ve středu okna.

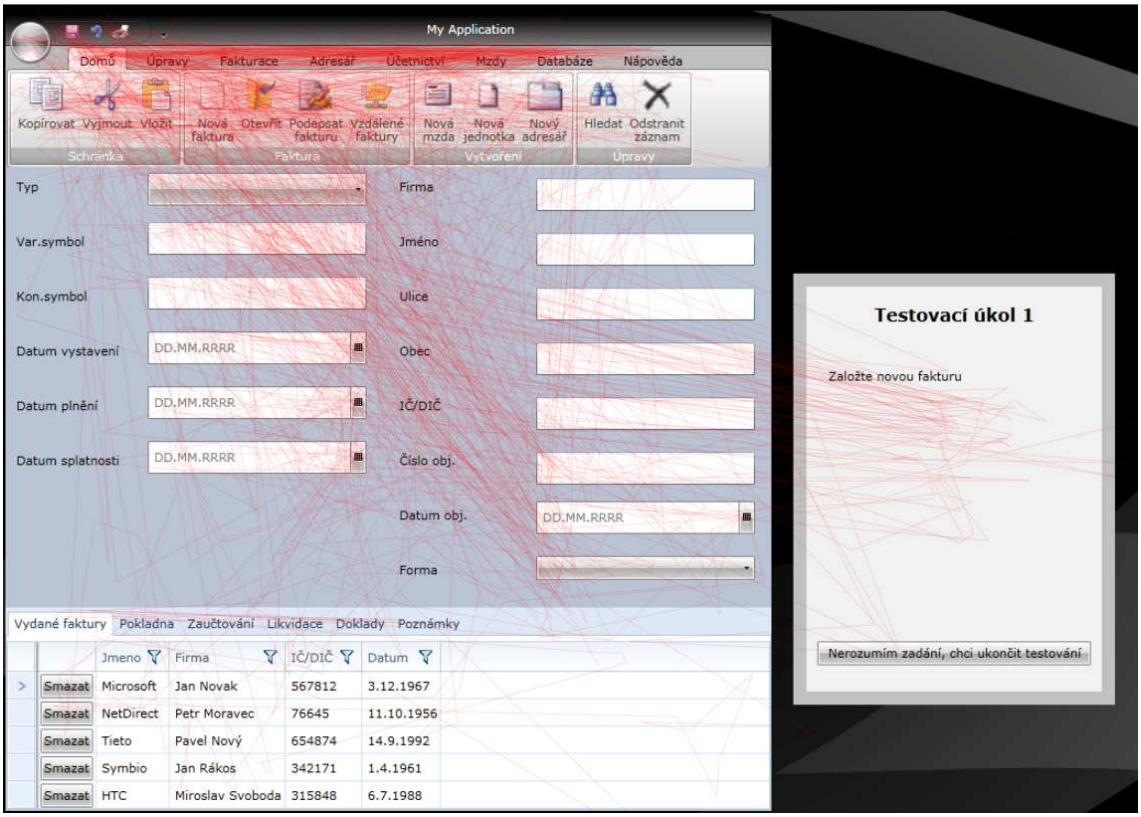
6.8.2.2. Test 3 – Postranní panel



Obrázek 6-12: Test 3 - Postranní panel - vyhodnocení pohybu myši pomocí čar

Z obrázku 6-12 lze vidět, že jsou čáry pohybů myší podobné jako v Testu 2. Přibyl však pohyb myší v postranním panelu a cesta mezi hlavním a postranním panelem. To je důvod, proč je u daného rozhraní vyšší vzdálenost uražená myší.

6.8.2.3. Test 4 – Ribbon



Obrázek 6-13: Test 4 - Ribbon - vyhodnocení pohybu myši pomocí čar

Jedním z důvodů pro vytvoření přesných obrazovek směrů pohybů myši bylo objasnění toho, proč průměrná vzdálenost při použití rozhraní Ribbon byla vyšší než u ostatních. Při srovnání obrazovek z pohybů myší a umístění červených teček Testu 2 a 4 bylo jejich rozmístění přibližně stejné. V obou případech byly horizontálně rozmístěny v horní části obrazovky. Tak proč ten rozdíl, když minimální uražená vzdálenost u rozhraní Ribbon se rovná u rozhraní Testu 2 průměrné uražené vzdálenosti? Při pozorování obrazovky rozhraní Ribbon, obrázek 6-13, si lze však všimnout zajímavé věci, a to že horizontální čáry jsou nejen v horní části s oušky karet, ale i v oblasti ikon. Přitom na rozdíl od rozhraní Testu 2, kdy se na ikonu musí najet myší, aby se ukázal popis, tohle není potřeba. Textový popis u ikon je zobrazen, a tak není třeba na ně najíždět. Jde tedy o pohyb, který uživatel provádí nevědomky, a který mu ulehčuje čtení a práci s rozhraním. Jeho pohyb očí následuje myší, aby mohl následně rychleji kliknout na požadovanou ikonu. Zajímavé je, že u rozhraní Testu 2 – Klasické menu nelze vidět pohyb, kdy by se uživatel pohyboval vertikálně ve vysunovacích nabídkách hlavní nabídky. Důvodem je nejspíše to, že stačí myší sjet z vysunovací nabídky a ta se hned zavře a on ji musí otevírat znova. Proto vždy jen nabídku otevře a výběr provádí nejdříve pouze očima. To u Ribbonu, díky jinému pojetí rozhraní, nehrozí. To by mohl být jeden z dalších důvodů větší dráhy myši. Dalším by mohla být menší přehlednost.

6.8.3 Test 5 – Práce s tabulkami

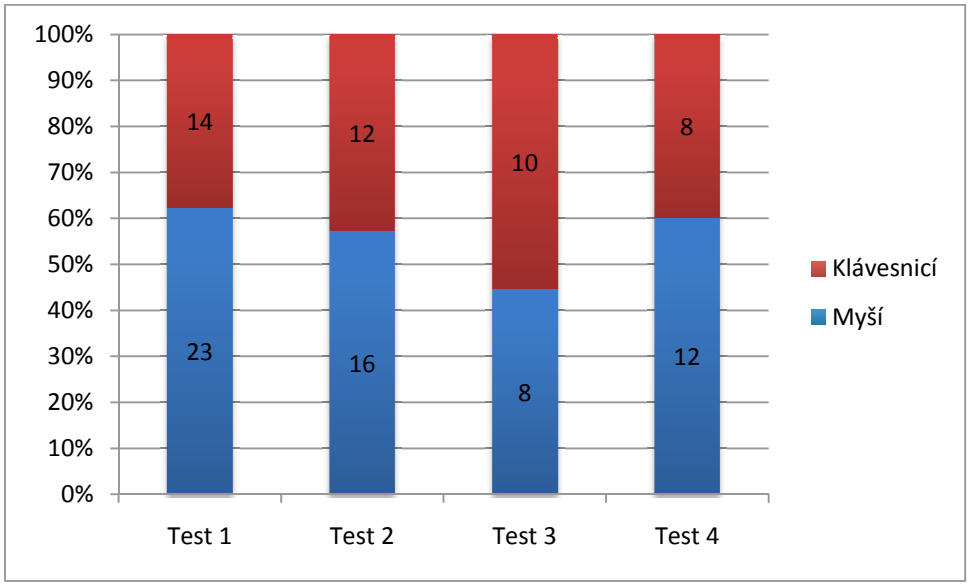
V posledním pátém testu byl pohyb myši největší. Velký vliv na tom mělo i to, že celkově celý test s tabulkami zabíral velkou plochu. Proto se části testujících s malým rozlišením nevešel do okna a oni museli skrolovat. Jenže z vyhodnocení testů práce s kolečkem myši lze vyčíst, že kolečko bylo použito jen v 37% případů. Ty ostatní proto museli myší vždy najet na skrolovací lišty. Zároveň testeři, kteří pro přesunutí prvků používali přetažení pomocí myší, měli výslednou uraženou vzdálenost určitě vyšší. Této možnosti však mnoho testerů nevyužilo.

6.9. Zadávávání data (Testy 1 – 4)

V testech 1-4 bylo několik testovacích případů, kdy měl tester za úkol zadat datum. Na výběr měl, zda ho zadá rovnou anebo zda dá přednost vybrat ho z kalendáře. U testu 1 byla jako zobrazená hodnota nastavené aktuální datum a tester měl vybrat datum následujícího dne. V testech 2-4 byla zobrazena maska ve tvaru „DD.MM.RRRR“ a tester měl zadat dnešní datum, zítřejší a datum za 10dní.

Z grafu 22 lze vyčíst, že mezi jednotlivými testy nejsou výraznější rozdíly. Takže použití přednastaveného data anebo masky v aplikaci nemá vliv na to, kolik uživatelů použije klávesnici anebo myš. Zároveň je vidět, že používání myši a klávesnice je poměrně vyrovnané, a proto je vhodné pokud aplikace podporuje oba způsoby zadávání data.

	Myší	Klávesnicí
Test 1	23	14
Test 2	16	12
Test 3	8	10
Test 4	12	8

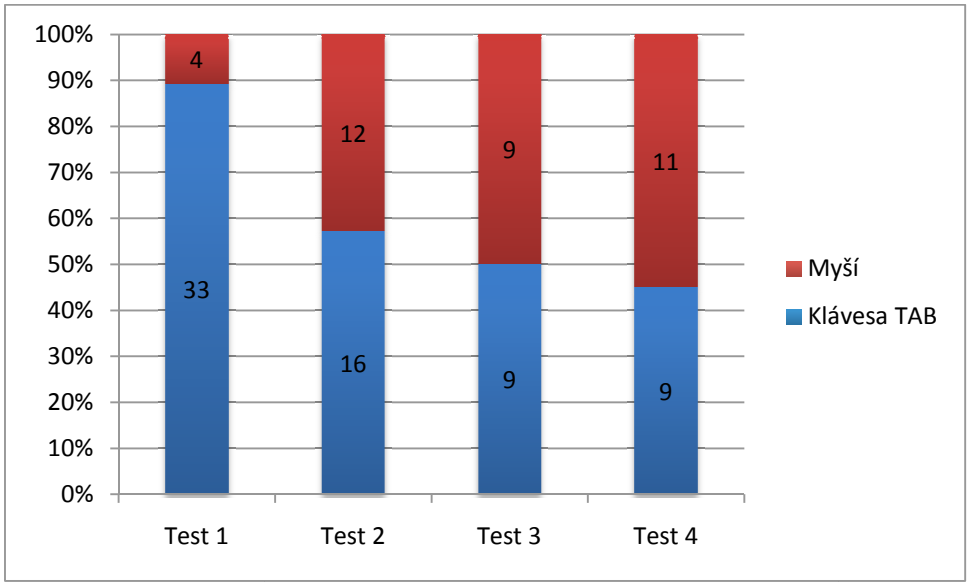


Graf 22: Způsob zadávání data

6.10. Procházení polí pomocí klávesy TAB

Zde se zjišťovalo, zda testeři pro přechod mezi poli použili klávesu `TAB` anebo přecházeli pomocí myši. U Testu 1 – Práce s formulářem plných 89% použilo pro procházení klávesu `TAB`. Tohle číslo je poměrně překvapivé, ale dá se vysvětlit tak, že celkem 78% všech testovaných uvedlo, že jsou zkušení až velmi zkušení uživatelé. V ostatních testech použití klávesy `TAB` nebylo natolik dominantní a bylo poměrově shodné s procházením pomocí klikání myší. Z toho tedy vyplývá, že pro procházení formulářů dávají uživatelé přednost klávese `TAB`. Zato u aplikací je používání klávesy `TAB` a klikání pomocí myší poměrově shodné.

	Klávesa TAB	Myší
Test 1	33	4
Test 2	16	12
Test 3	9	9
Test 4	9	11



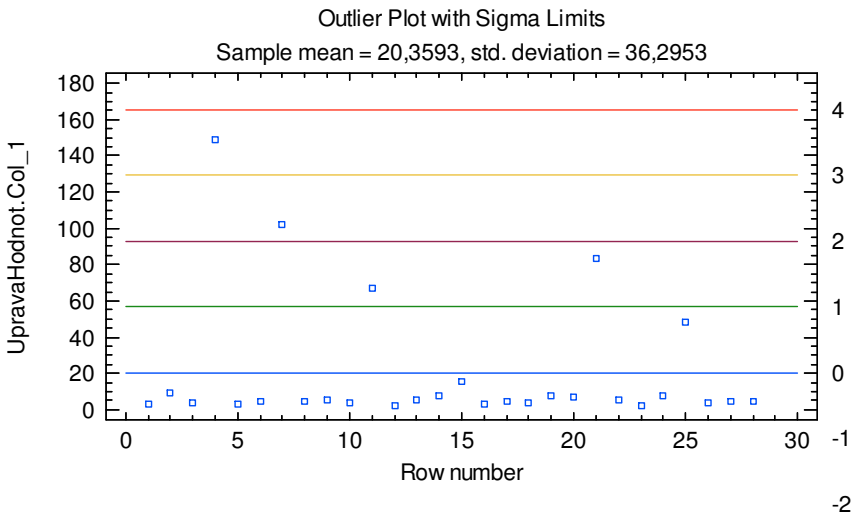
Graf 23: Procházení polí pomocí klávesy TAB

6.11. Porovnání časů jednotlivých testovacích úkolů v testu 2-4

Testovací úkoly v testech 2-4 byly shodné. V každém testu tedy bylo 8 shodných úkolů. Proto se porovnaly časy jednotlivých testovacích úkolů, aby se zjistilo, které řešení je lepší z pohledu rychlosti.

Před samotným zpracováním dat však bylo zjištěno, že několik dat z testů vykazuje velké odchylky a mohlo by se jednat o odlehlé hodnoty. Jelikož vykonaných testů je poměrně malý počet, tak by tyto odlehlé hodnoty mohly způsobit velké odchylky v měření. Proto bylo pro jejich odhalení použito programu Statgraphics a jeho funkce `Outlier Identification` [33].

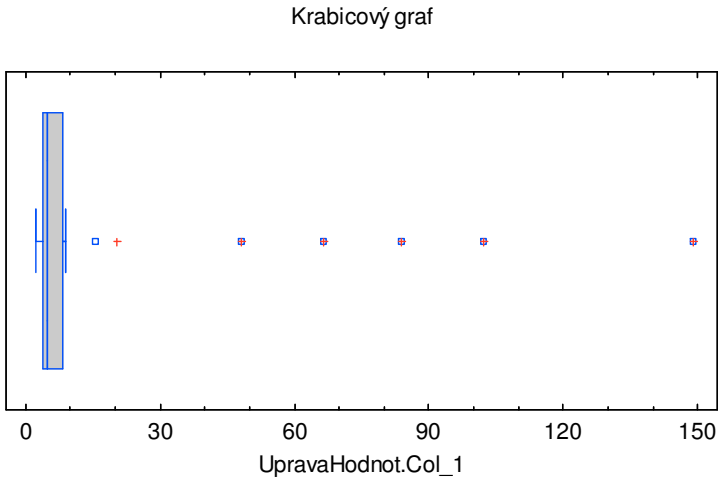
Ta slouží právě k odhalení odchylek v měření. Pro příklad je uveden výsledek testu pro první testovací úkol testu 2 – Klasické menu. Střední hodnota (mean) je v tomto případě 20,35 a směrodatná odchylka (std. Deviation) 36,29.



Graf 24: Odlehlé hodnoty, Test 2 - první testovací případ

Na grafu 24 je tedy hezky vidět, že většina hodnot je pod 20 sekund, ale pár hodnot je mnohem vyšších. Osa napravo ukazuje násobky směrodatné odchylky. Hodnota 148 tedy překračuje směrodatnou odchylku více než 3x.

Pro učení odlehlých hodnot lze použít i krabicový graf. V grafu 25 je odchylka odlehlých hodnot velmi patrná.



Graf 25: Krabicový graf, Test 2 - první testovací případ

Bylo tedy zjištěno, že naše měření obsahuje často výrazně odlehlé hodnoty. Jelikož naše sady dat jsou poměrně malé, je riziko, že odlehlé hodnoty budou výsledkem nedostatečného počtu testů. Zároveň pro odebrání dat by měl být znám důvod, který vedl ke vzniku takto odlehlé hodnoty. V našem případě lze však jen těžko určit, co bylo daným důvodem. Je možné, že uživatel spustil test a mezitím se věnoval jiné činnosti než vykonávání testu.

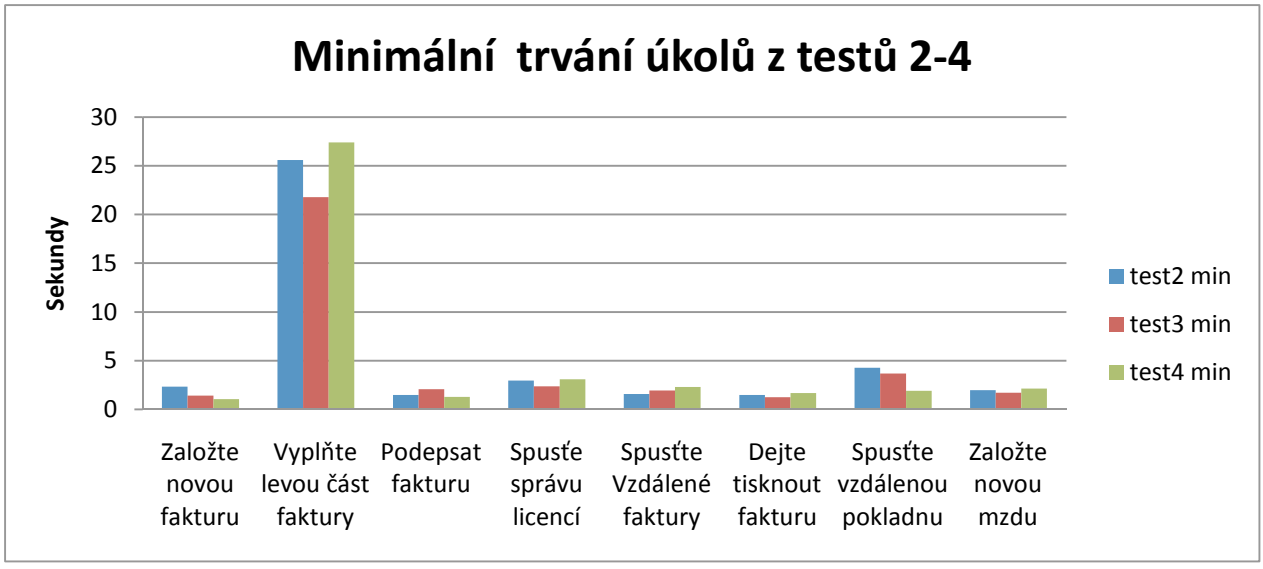
Nakonec bylo rozhodnuto, že by některé odlehlé hodnoty způsobily velké zkreslení výsledků a proto by měly být odstraněny. Odstraněny byly však jen velmi odlehlé hodnoty a ty, které překračovaly směrodatnou odchylku minimálně trojnásobně. Celkem bylo odstraněno 11 hodnot.

Pro posouzení byly vybrány parametry průměrný čas, medián, maximální a minimální čas:

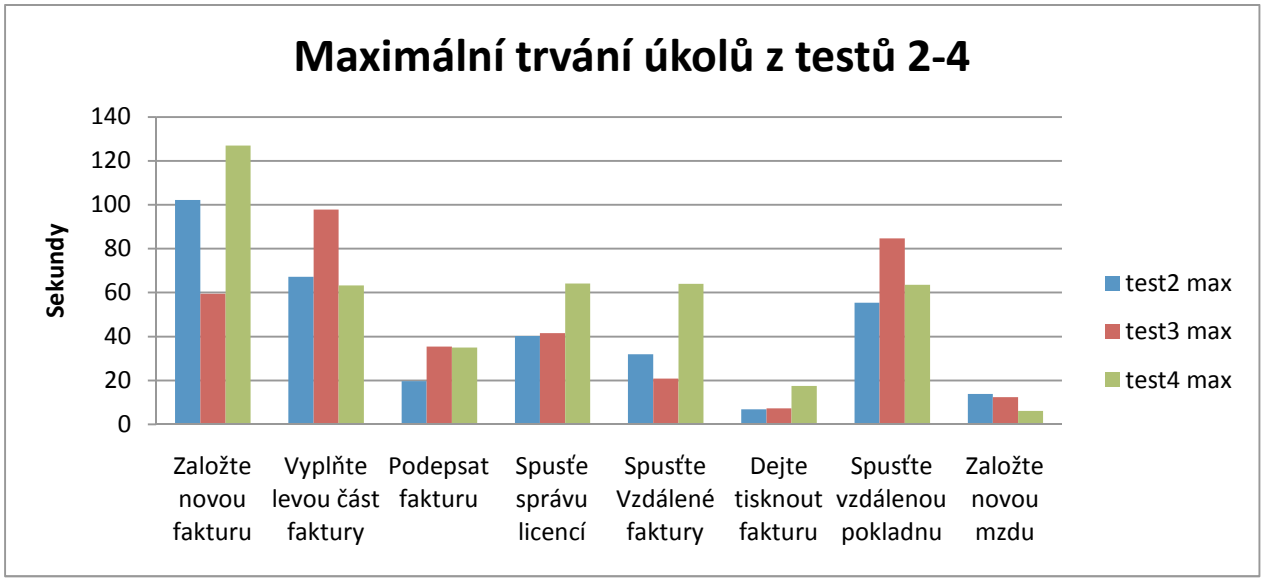
Průměrný čas nám říká, za jaký průměrný čas byl test dokončen. Je však ovlivněn minimálními a maximálními hodnotami.

Medián je hodnota, která dělí řadu podle velikosti seřazených výsledků na dvě stejně početné části. Platí, že nejméně 50 % hodnot je menších nebo rovných a nejméně 50 % hodnot je větších nebo rovných mediánu. Výhodou je, že není ovlivněn extrémy jako průměr.

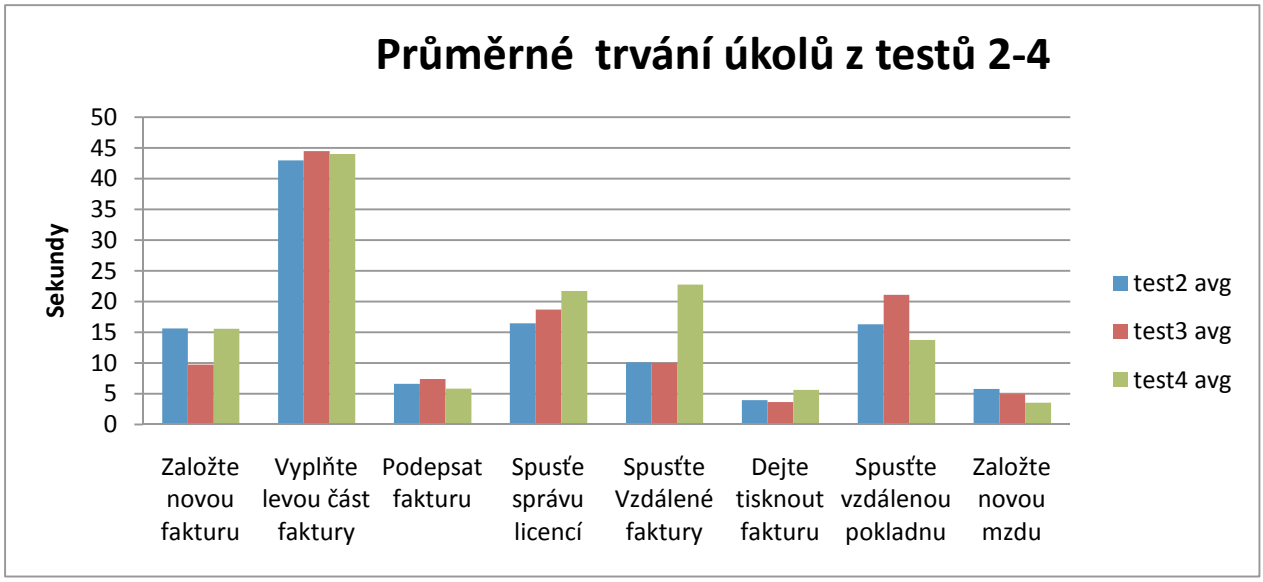
Maximální a minimální hodnota nám označuje extrémy. Takže jak nejrychleji a nejpomaleji by trvalo vyplnit daný test v ideálním případě.



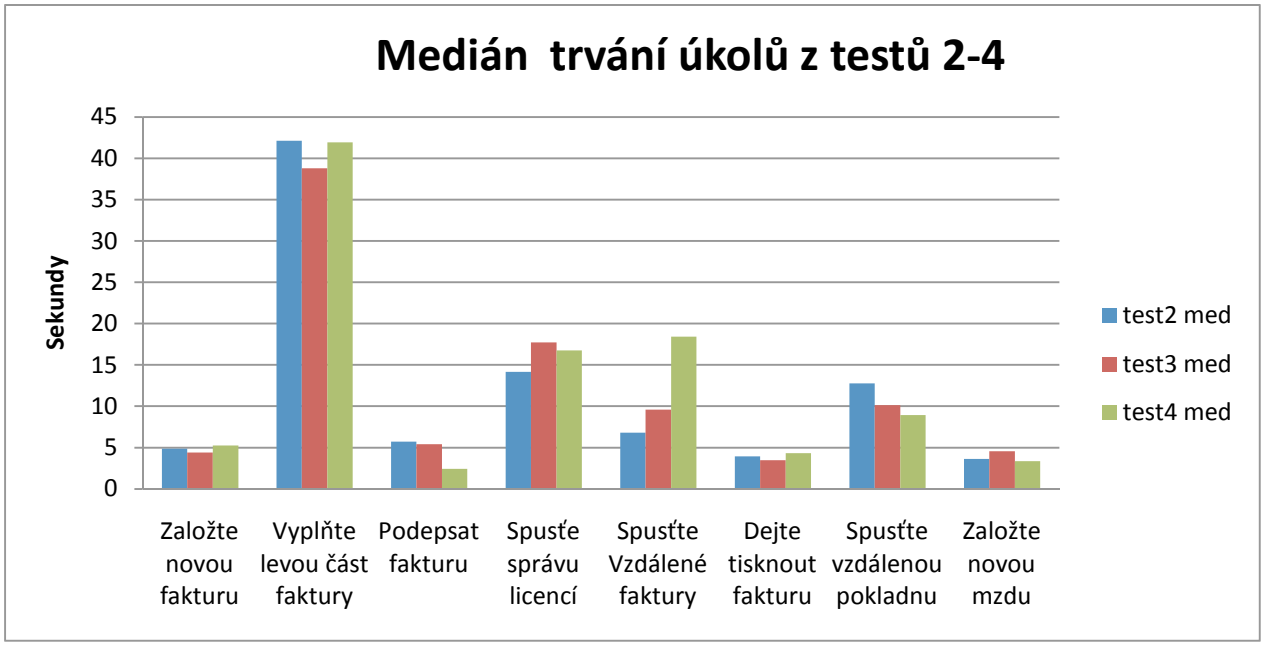
Graf 26: Minimální trvání úkolů z testu 2-4



Graf 27: Maximální trvání úkolů z testu 2-4



Graf 28: Průměrné trvání úkolů z testu 2-4

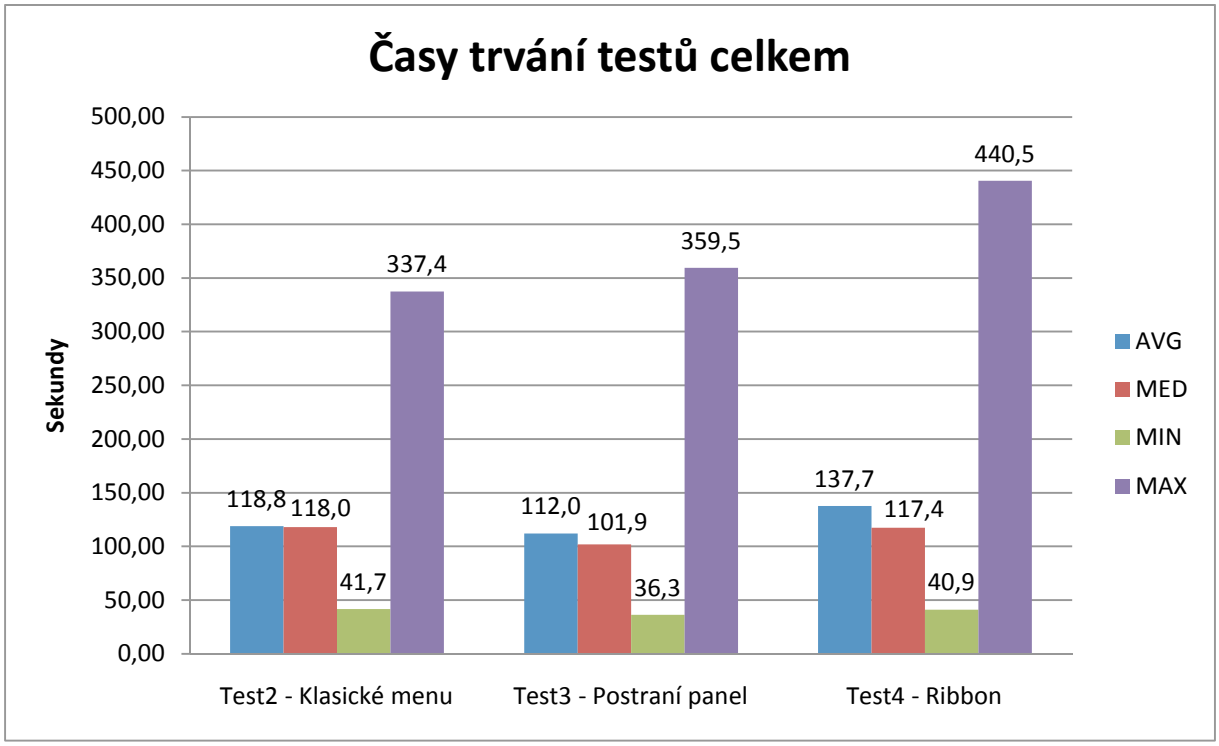


Graf 29: Medián trvání úkolů z testu 2-4

6.11.1 Závěry celkový čas

Pro porovnání celkových časů byl použit graf 30. Ten byl vytvořen tak, že se vzaly hodnoty celkového trvání jednotlivých testů a odstranily se ty, které obsahovaly odlehlé hodnoty nalezené pomocí funkce `Outlier Identification`. Pokud by byly brány v potaz průměrné hodnoty, tak by nejrychlejším rozhraním obecně bylo to, které bylo použito v Testu 3 neboli postranní panel. Na druhém místě by bylo klasické menu a jako třetí Ribbon. Při porovnávání podle průměrů však dochází k výrazným zkreslením díky extrémům. Vhodnější proto bude pro porovnání použít medián. Při porovnání mediánů byl opět nejrychlejší Test 3 – Postranní panel. Tentokrát však Ribbon skončil jako druhý a těsně za ním se pak umístilo rozhraní testu 2 – Klasické menu.

Při porovnání rozhraní z pohledu extrémů by bylo nejrychlejším rozhraním rozhraní z Testu 3 - Postranní panel, následováno Ribbonem a Klasickým menu. Nejdelší čas by naopak zabralo rozhraní Ribbon, za ním následuje Postranní panel a Klasické menu.



Graf 30: Souhrnný graf trvání testů 2-4

Jednotlivé časy testů byly poté porovnány pomocí neparametrického Kruskal-Wallisova testu. Tento test byl vybrán, jelikož nevyžaduje normalitu rozdělení dat. Tuhle podmínku totiž výběr nesplňuje. Data však musí splňovat podmínku homoskedasticity, identických rozptylů.

První byly porovnány průměrné hodnoty jednotlivých výběrů. Nejdříve byla ověřena podmínka homoskedasticity. Ta byla potvrzena, a tak se pokračovalo v Kruskal-Wallisově testu.

Jako nulová hypotéza byla zvolena možnost, že celkové hodnoty jednotlivých testů se neliší. P-Value vyšlo, že se rovná 0,324629, což je více jak 0,05. Takže nebyla zamítnuta nulová hypotéza. Dá se tedy říct s 95% spolehlivostí, že použitý typ rozhraní nemá významný vliv na celkový čas vykonání testu.

Stejný test byl poté proveden i pro hodnoty průměrů jednotlivých testovacích úkolů. Jako nulová hypotéza byla zvolena možnost, že hodnoty jednotlivých testovacích úkolů v různých testech se neliší. Testovací data splňovala podmínku homoskedasticity. P-Value vyšlo rovno 0,968022, což je více jak 0,05. Takže nebyla zamítnuta nulová hypotéza. Dá se tedy říct s 95% spolehlivostí, že použitý typ rozhraní nemá významný vliv na průměrný čas vykonávání jednotlivých testovacích úkolů jednotlivých testů. Stejný test byl poté proveden i pro hodnoty mediánů. Dané hodnoty taktéž splňovaly podmínku homoskedasticity a P-Value opět vyšlo větší jak 0,05, přesně 0,970446. Proto i v tomhle případě se dá říci s 95% spolehlivostí, že použitý typ rozhraní nemá významný vliv na mediánový čas vykonávání jednotlivých úkolů.

Z grafů a pomocí Kruskal-Wallisova testů byl vyvozen závěr, že sice rozhraní testu 3, neboli Postranní panel byl při vykonávání testovacích úkolů nejrychlejší, ale s přihlédnutím k

výsledkům Kruskal-Wallisových testů a minimálním rozdílům řádově v jednotkách procent se dá říci, že jednotlivá použitá rozhraní jsou do času vykonávání jednotlivých úkolů víceméně shodná, a neexistuje mezi nimi významný rozdíl.

6.11.2 Závěry jednotlivých úkolů

Úkol 1: Založte novou fakturu

V tomto úkolu šlo vesměs jen o kliknutí na položku `Nová faktura`. Při porovnání průměrů byl tento úkol nejrychleji splněn při použití postranního panelu, v průměru za 10 sec. Klasické menu a Ribbon byli o 5 sec pomalejší. Při porovnání mediánů však mezi rozhraními není časově vesměs žádný rozdíl.

Jednotlivé časy testovacího úkolu 1 byly také porovnány pomocí analytické metody ANOVA. Anova je analýza rozptylu středních hodnot nezávislých náhodných výběrů. Požadavkem je normální rozdělení dat a homoskedasticita (identické rozptyly).

Nejdříve byla ověřena podmínka homoskedasticity. Ta byla potvrzena. Poté byla provedena ANOVA. Jako nulová hypotéza byla zvolena možnost, že střední hodnoty časů testovacího úkolu 1 se při různých rozhraních neliší. P-Value vyšlo, že se rovná 0,7123, což je více jak 0,05. Nebyla tedy zamítnuta nulová hypotéza. Dá se tedy říct s 95% spolehlivostí, že neexistuje statisticky významný rozdíl mezi středními hodnotami jednotlivých rozhraní u testovacího úkolu 1.

Zajímavostí je velký rozdíl mezi výsledky průměrů a výsledky mediánů. Lze vidět, že velké procento testerů při prvním úkolu prozkoumávalo aplikaci a jen část ji začala hned vyplňovat.

Úkol 2: Vyplňte levou část faktury

V tomto úkolu šlo o vyplnění faktury (formuláře) a poté zvolení `Uložit`. Tedy zde použité rozhraní hrálo velmi malou roli, a šlo jen o to, jak jsou testeři schopni rychle vyplnit formulář. Při porovnání průměrů byl tento úkol splněn přibližně u všech testů stejně rychle. I při porovnání mediánů mezi rozhraními nebyl časově vesměs žádný rozdíl.

K analýze byla poté použita i analytická metoda ANOVA. Postup byl obdobný, jako je popsán u prvního testovacího úkolu. Vyšlo, s 95% spolehlivostí, že neexistuje statisticky významný rozdíl mezi středními hodnotami jednotlivých rozhraní u testovacího úkolu 2.

Úkol 3: Podepsat fakturu

V tomto úkolu šlo o kliknutí na příkaz `Podepsat fakturu`. Při porovnání průměrů byl tento úkol splněn průměrně u všech testů stejně rychle, rozdíl byl v rámci jedné sekundy. Při porovnání mediánů však byl Ribbon nejrychlejší, a to až dvojnásobně.

K analýze byla poté použita i analytická metoda ANOVA. Postup byl obdobný, jako je popsán u prvního testovacího úkolu. Vyšlo, s 95% spolehlivostí, že neexistuje statisticky významný rozdíl mezi středními hodnotami jednotlivých rozhraní u testovacího úkolu 3.

Úkol 4: Spustit správu licencí

V tomto úkolu šlo o kliknutí na příkaz `Spustit správu licencí`. Při porovnání průměrů byl tento úkol splněn nejrychleji při použití klasického menu, následován postranním panelem a Ribbonem. Při porovnání mediánů bylo klasické menu opět nejrychlejší, Ribbon a postranní panel dopadly přibližně stejně.

K analýze byla poté použita i analytická metoda ANOVA. Postup byl obdobný, jako je popsán u prvního testovacího úkolu. Vyšlo, s 95% spolehlivostí, že neexistuje statisticky významný rozdíl mezi středními hodnotami jednotlivých rozhraní u testovacího úkolu 4.

Úkol 5: Spustit vzdálenou fakturu

V tomto úkolu šlo o kliknutí na příkaz `Spustit vzdálenou fakturu`. Při porovnání průměrů byl tento úkol splněn stejně rychle při použití klasického menu i postranního panelu. Ribbon byl dvakrát pomalejší. Při porovnání mediánů bylo klasické menu nejrychlejší, následováno postranním panelem a Ribbonem, který byl ale 3x pomalejší než klasické menu.

K analýze byla poté použita i analytická metoda ANOVA. Postup byl obdobný, jak je popsán u prvního testovacího úkolu. Tentokrát nám vyšlo, s 95% spolehlivostí, že existuje statisticky významný rozdíl mezi středními hodnotami jednotlivých rozhraní u testovacího úkolu 5. Proto jsme poté použili post-hoc analýzu a zjistili, že rozhraní klasické menu a postranní panel by se dalo považovat za shodné, ale Ribbon nikoliv.

Úkol 6: Tisk faktury

V tomto úkolu šlo o kliknutí na příkaz `Tisk`. Při porovnání průměrů byl tento úkol splněn stejně rychle při použití klasického menu a postranního panelu, Ribbon byl o něco pomalejší. Při porovnání mediánů byla rozhraní přibližně stejně rychlá.

K analýze byla poté použita i analytická metoda ANOVA. Postup byl obdobný, jako je popsán u prvního testovacího úkolu. Vyšlo, s 95% spolehlivostí, že neexistuje statisticky významný rozdíl mezi středními hodnotami jednotlivých rozhraní u testovacího úkolu 6.

Úkol 7: Spusťte vzdálenou pokladnu

V tomto úkolu se jednalo o kliknutí na příkaz `Vzdálená pokladna`. Při porovnání průměrů byl tento úkol splněn stejně rychle při použití Ribbonu, následován klasickým menu a postranním panelem. Při porovnání mediánů byl opět nejrychlejší Ribbon, za ním klasické menu a postranní panel.
K analýze byla poté použita i analytická metoda ANOVA. Postup byl obdobný, jako je popsán u prvního testovacího úkolu. Vyšlo, s 95% spolehlivostí, že neexistuje statisticky významný rozdíl mezi středními hodnotami jednotlivých rozhraní u testovacího úkolu 7.

Úkol 8: Založte novou mzdu

V tomto úkolu se mělo kliknout na příkaz `Nová mzda`. Při porovnání průměrů byl tento úkol splněn stejně rychle při použití Ribbonu, následován postranním panelem a klasickým menu. Při

porovnání mediánů byl opět nejrychlejší Ribbon, následován klasickým menu a postranním panelem.

K analýze byla poté použita i analytická metoda ANOVA. Postup byl obdobný, jako je popsán u prvního testovacího úkolu. Vyšlo, s 95% spolehlivostí, že neexistuje statisticky významný rozdíl mezi středními hodnotami jednotlivých rozhraní u testovacího úkolu 8.

7. Sledování závislostí

Cílem těchto testů bylo sledovat vztahy mezi dvěma veličinami a najít případné závislosti. Jedním ze sledovaných vztahů mělo být, zda a jak schopnosti zadané testery ovlivňují způsob ovládání aplikace. Ale pro nízkou diverzitu uvedených schopností, kdy jen 8% testovaných uvedlo své zkušenosti jako nezkušení anebo začátečníci, bylo od těchto testů upuštěno. Díky tomu, že většina zkoumaných dat byla nekategoriálních, pro sledování případných vztahů byla použita lineární regrese.

7.1. Závislost použití myši a čas potřebný k vykonání testů

V tomto případě mělo být ověřeno, zda míra využívání myši má vliv na čas vykonání testu. Pro vyhodnocení lineární regrese byl opět použit program Statgraphics. Jako nezávislá proměnná byla zvolena uražená vzdálenost a jako závislá proměnná čas vykonání testu.

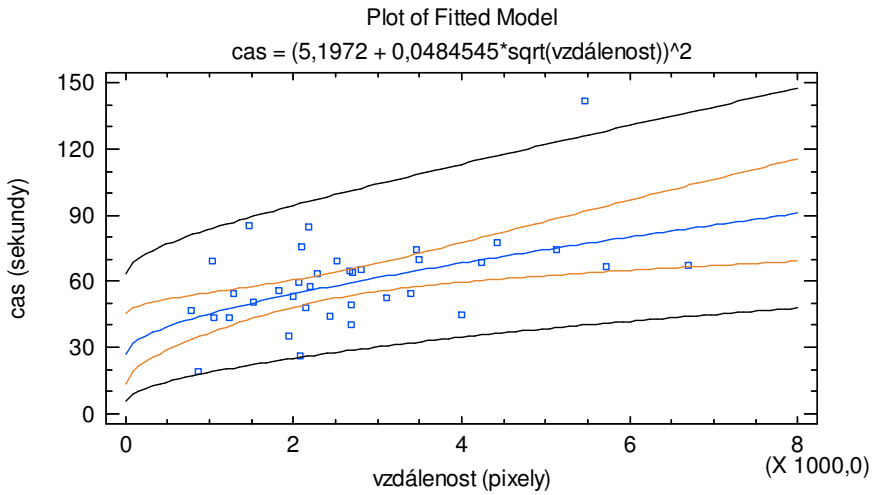
Před každým testem bylo prozkoumáno korelační pole, zda neobsahuje tzv. vlivné body. Ty by mohly značit extrémy a nebo odlehlé body. Vlivné body mohou mít silný vliv na odhadovanou regresní funkci. Proto byly některé z nich odstraněny, ale bylo postupováno obezřetně, a vyřazeny byly jen ty opravdu velmi významné. Jejich vznik mohlo zapříčinit to, že se testeři nevěnovali testování, ale jen zkoumali dané rozhraní, což se odrazilo ve výsledném čase.

7.1.1 Test 1 – Práce s formuláři

Nejdříve byla zobrazena rovnice vyrovnávací přímky a poté provedeny dílčí t-testy pro testování významnosti parametrů vyrovnávací přímky. V obou případech vyšly hodnoty P-Value menší než 0,05, tedy v obou případech se H_0 zamítá. Neboli ani jeden z parametrů a, b nelze z modelu vypustit. Poté bylo pomocí tabulky ANOVA ověřeno, zda existuje nějaká závislost mezi hodnotami. Jelikož P-Value vyšlo menší než 0,05, existuje mezi veličinami závislost.

Poté byly vyhodnoceny předpoklady pro použití lineárního regresního modelu. Byly otestovány rezidua, zda splňují podmínky normality a nulovosti středních hodnot reziduí. Ty byly potvrzeny.

Následně byla ověřena kvalita modelu. Index determinace při použití lineárního modelu byl pouze 18% a korelační koeficient byl 0,4275, což značí poměrně slabý vztah mezi veličinami. Model byl poté změněn za vhodnější na `double square root`, ale i tak index terminace byl jen 24,5%.

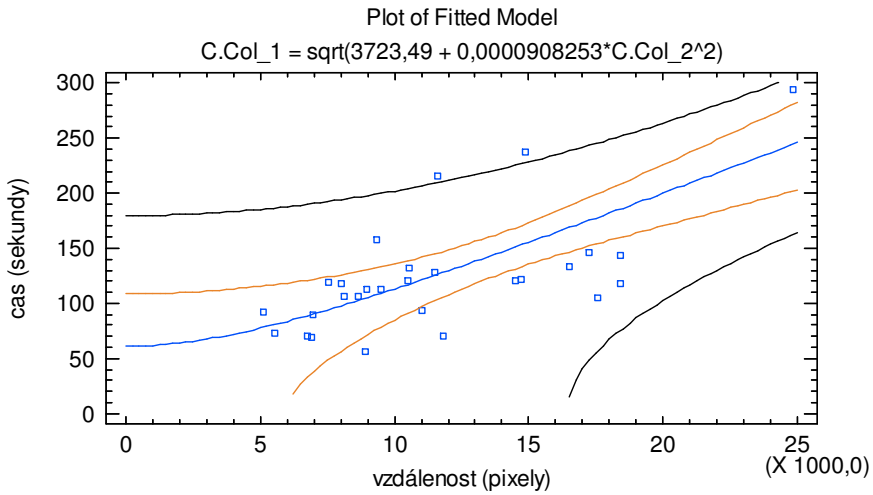


Graf 31: Závislost myš/čas – Test 1

Z testu se dá tedy vyvodit, že existuje závislost mezi uraženou vzdáleností a časem trvání testu. Tato závislost však není příliš silná, a i z grafu 31 lze vyčíst, že při stoupající vzdálenosti neroste čas vykonání testu nijak významně.

7.1.2 Test 2 – Klasické menu

U testu dva bylo postupováno obdobně jak u testu jedna. Takže ve zkratce, byla opět potvrzena závislost i všechny předpoklady. Index determinace při použití lineárního modelu byl 40% a korelační koeficient byl 0,6391, což značí již poměrně silnější vztah mezi veličinami, než u prvního testu. Model byl poté změněn za vhodnější na `double square`, a tak index determinace byl již 46% a korelační koeficient byl 0,6772.

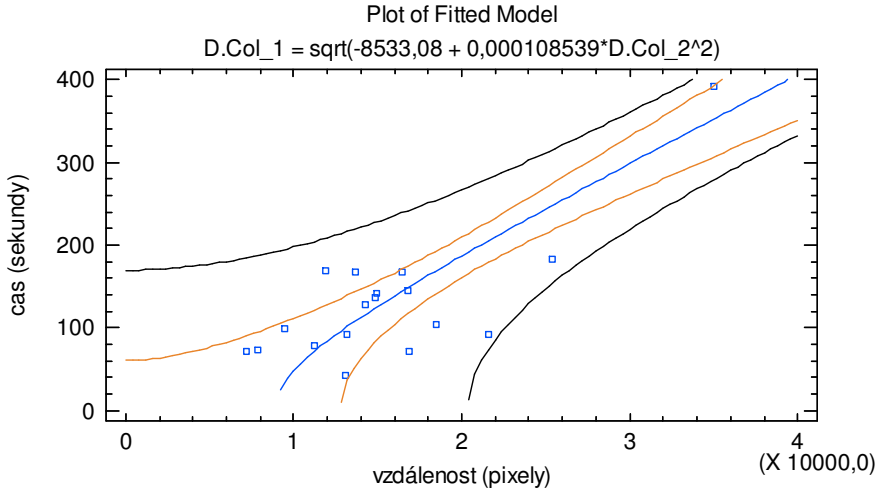


Graf 32: Závislost myš/čas – Test 2

Z testu se dá tedy vyvodit, že existuje závislost mezi uraženou vzdáleností a časem trvání testu. Tahle závislost je již poměrně silná, ale čas se stoupající vzdáleností neroste nijak strmě.

7.1.3 Test 3 – Postranní panel

Opět byla potvrzena závislost i všechny předpoklady. U tohoto testu se však silně projevilo, jak je ošidné pracovat s odlehlými body. Se všemi body byl dosažen index determinace až 78% a korelační koeficient 0,8890. Pokud však byl odstraněn jeden odlehlý bod, klesl index determinace až k 18%. Po úvaze byl poté daný odlehlý bod opět vrácen. Na grafu 33 je jasně vidět, jak bod napravo ovlivňuje celý graf.

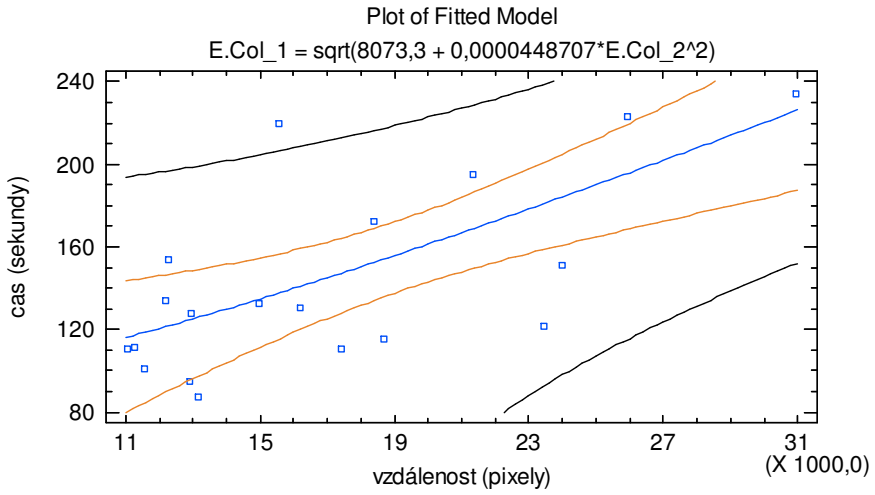


Graf 33: Závislost myš/čas – Test 3

Závěrem lze tedy říct, že existuje i v tomto případě závislost mezi uraženou vzdáleností a časem trvání testu. Ale jak silná je tato závislost říci nelze, protože díky malému počtu hodnot lze těžko vyhodnotit, zda bod úplně napravo lze považovat za chybu anebo za extrém, který by však neměl být odstraněn.

7.1.4 Test 4 - Ribbon

Bylo postupováno obdobně jak u předešlých testů. Byla opět potvrzena závislost i všechny předpoklady. Index determinace při použití `double square` modelu byl 48% a korelační koeficient 0,6932, což značí již poměrně silný vztah mezi veličinami.

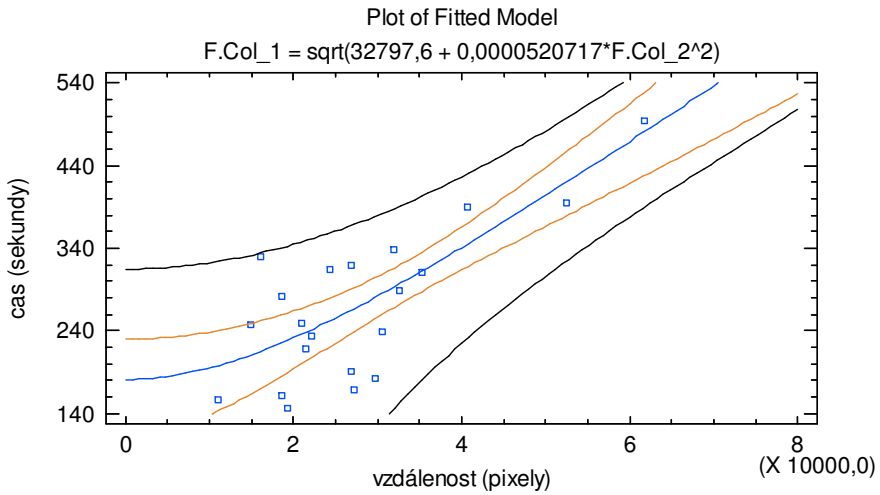


Graf 34: Závislost myš/čas – Test 4

Z testu se dá tedy vyvodit, že existuje závislost mezi uraženou vzdáleností a časem trvání testu. Tato závislost je již poměrně silná a čas se stoupající vzdáleností roste už poměrně významně.

7.1.5 Test 5 – Práce s tabulkami

Postup byl obdobný jako u předešlých testů. Opět byla potvrzena závislost i všechny předpoklady. Index determinace při použití `double square` modelu byl 71% a korelační koeficient byl 0,8435, což již značí silný vztah mezi veličinami.



Graf 35: Závislost myš/čas – Test 5

Z testu se dá tedy vyvodit, že existuje závislost mezi uraženou vzdáleností a časem trvání testu. Ze všech provedených testů je tato závislost nejsilnější, a dá se tedy říct, že se stoupající uraženou vzdáleností roste poměrně významně čas vykonání testu.

7.1.6 Zhodnocení

Při zhodnocení testů závislostí vyšlo najevo, že ve všech případech existuje závislost mezi uraženou vzdáleností a časem vykonání testu, a to taková, že čím více tester pohyboval myší, tím déle mu trvalo splnění testu. U většiny případů jde však o závislost poměrně nízkou. Výraznější závislost lze vidět pouze u Testu 5 Práce s tabulkami. Závěr je to poměrně logický, neboť zvýšený pohyb myší v Testu 2-4 může značit, že uživatel hledal v menu příkazy, a tak mu vykonání testu trvalo déle. U Testu 5 může zvýšený pohyb myší značit to, že uživatel nepoužíval pokročilé prvky jako filtry, ale vybíral prvky „ručně“, a proto byl čas i pohyb myši vyšší.

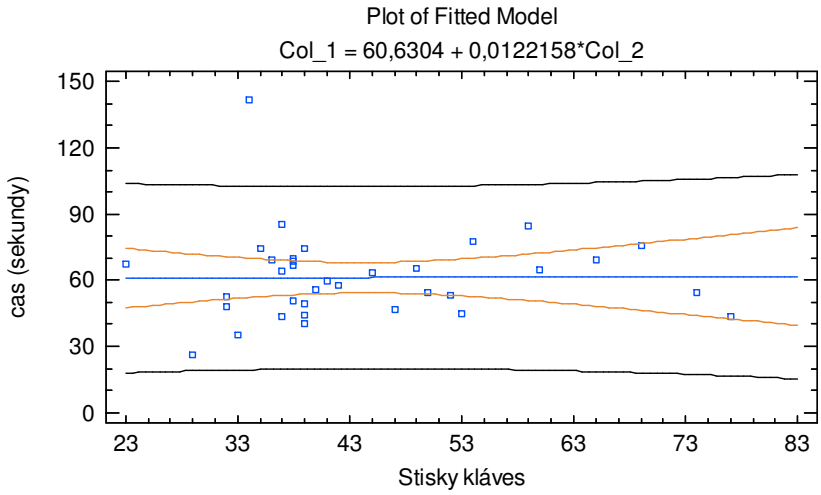
7.2. Závislost použití klávesnice a rychlosti vykonání testů

V tomto případě mělo být ověřeno, zda míra využívání klávesnice má vliv na čas potřebný k vykonání testu. Jako nezávislá proměnná byl zvolen počet stisknutých kláves a jako závislá proměnná čas vykonání testu.

Před každým testem bylo, stejně jak v předchozím případě, prozkoumáno korelační pole, zda neobsahuje tzv. vlivné body.

7.2.1 Test 1 – Práce s formuláři

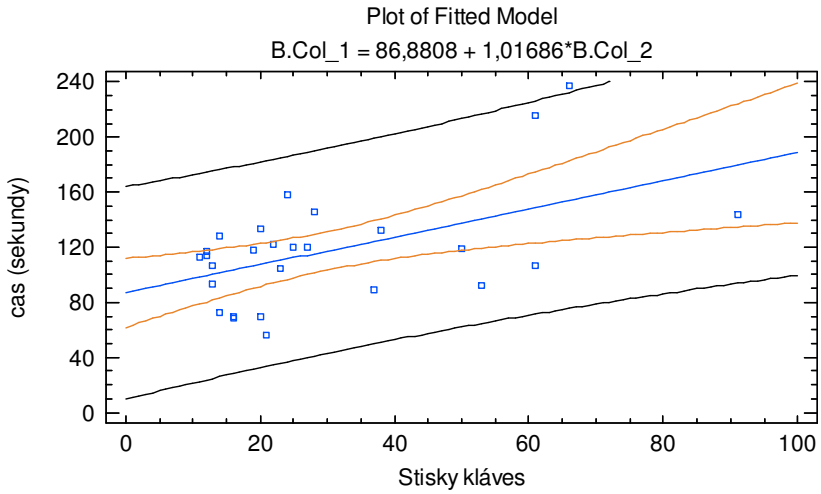
Postup byl obdobný jak v předešlých testech. V tomto případě však bylo zjištěno pomocí tabulky ANOVA, že neexistuje závislost mezi hodnotami. P-Value nám totiž vyšlo větší než 0,05, přesněji 0,9639. Z toho vyplývá, že při práci s formuláři, to v jakém rozsahu používají uživatelé klávesnici, nemá vliv na čas vykonání testu. Lze to vyčíst i z grafu 36.



Graf 36: Závislost stisky kláves/čas – Test 1

7.2.2 Test 2 – Klasické menu

Postup byl obdobný jako u předešlých testů. Opět byla potvrzena závislost i všechny předpoklady. Index determinace při použití lineárního modelu byl 67% a korelační koeficient byl 0,8230, což značí silný vztah mezi veličinami. Při odstranění jedné odlehlé hodnoty však index determinace klesl na 26% a korelační koeficient na 0,5165, což značí slabý vztah mezi veličinami. Daný test s odlehlou hodnotou byl prozkoumán a bylo zjištěno, že stisknuté klávesy byly jen změť znaků, které nesouvisí se zadáním testu.

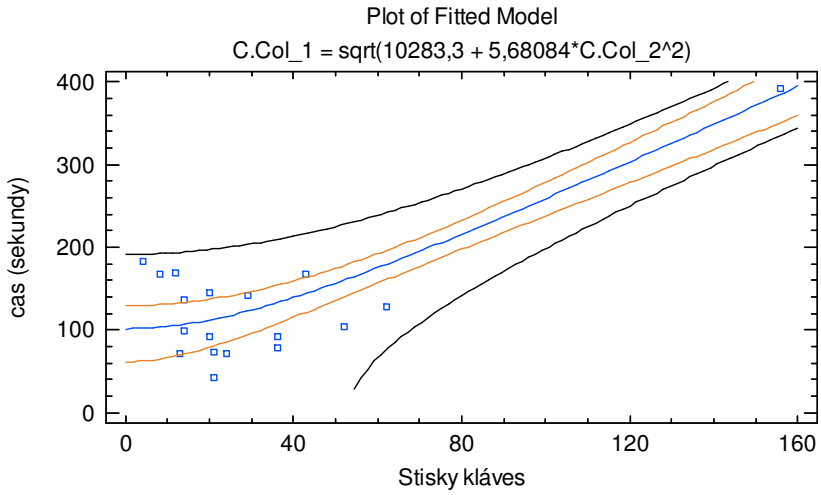


Graf 37: Závislost stisky kláves/čas – Test 2

Závěrem se tedy dá říct, že mezi používáním klávesnice a rychlostí vykonání testu existuje závislost, avšak poměrně slabá.

7.2.3 Test 3 – Postranní panel

U Testu 3 bylo postupováno obdobně jako u předešlých testů. Bez odstranění jakýchkoliv odlehlých měření byl index determinace při použití `double square` modelu vysokých 88% a korelační koeficient byl 0,9388, což by značilo velmi silný vztah mezi veličinami. Bylo však rozhodnuto o odstranění jedné odlehlé hodnoty, která několikanásobně překračovala jak čas, tak počet stisků kláves ostatních testů, viz Graf 38. Při zkoumání stisků kláves daného testu vyšlo najevo, že se jedná pouze o stisky numerických kláves.

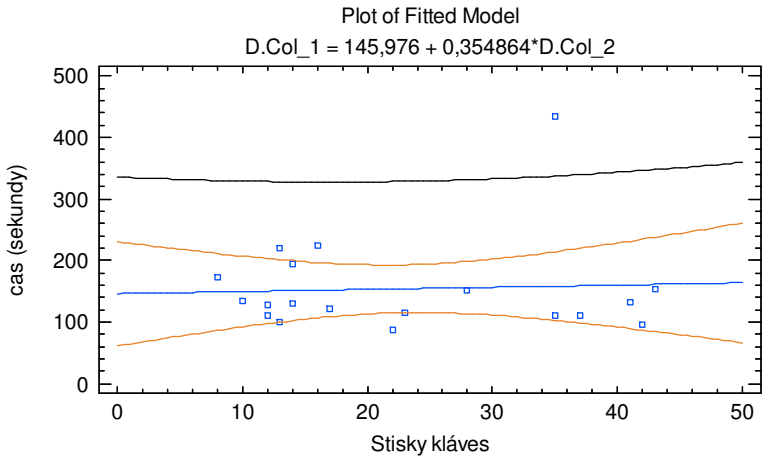


Graf 38: Závislost stisky kláves/čas – Test 3

Po odstranění dané hodnoty poté pomocí tabulky ANOVA bylo zjištěno, že neexistuje závislost mezi hodnotami. P-Value totiž vyšlo větší než 0,05, přesněji 0,6011. Takže u Testu 3 nebyla prokázána žádná závislost mezi používáním klávesnice a rychlostí vykonání testu.

7.2.4 Test 4 – Ribbon

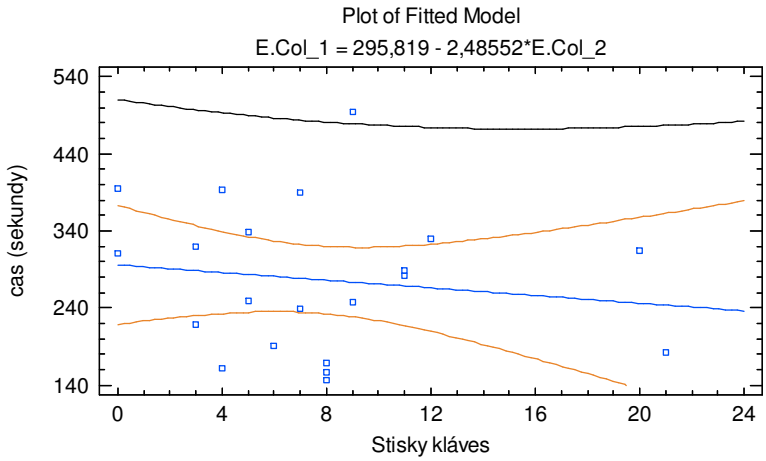
U Testu 4 se opakovala obdobná situace jako u testů předešlých. Pomocí tabulky ANOVA bylo zjištěno, že neexistuje závislost mezi hodnotami. P-Value totiž vyšlo větší než 0,05, přesněji 0,8225. Z toho vyplývá, že při práci s formuláři rozsah používání klávesnice nemá vliv na čas vykonání testu, viz Graf 39.



Graf 39: Závislost stisky kláves/čas – Test 4

7.2.5 Test 5 – Práce s tabulkami

I u Testu 5 se opakovala obdobná situace jak u testů předešlých. Pomocí tabulky ANOVA bylo zjištěno, že neexistuje závislost mezi hodnotami. P-Value totiž vyšlo větší než 0,05, přesněji 0,5389. Pokud by přesto byla daná závislost uznána, tak pokrytí a korelační koeficient by byly tak nízké, že by stejně nic neprokazovaly. Z toho vyplývá, že při práci s tabulkami rozsah používání klávesnice nemá vliv na čas vykonání testu. Viz Graf 40.



Graf 40: Závislost stisky kláves/čas – Test 5

7.2.6 Zhodnocení

Po zhodnocení testů závislostí vyšlo najevo, že ve většině případů neexistuje závislost mezi rozsahem používání klávesnice a rychlostí vykonání testů. Závislost byla prokázána jen u Testu 2 – Postranní panel, a to ještě poměrně slabá.

8. Zhodnocení výsledků

Předem bych chtěl ještě jednou zmínit, že mnou dosažené závěry, stejně jako všechny závěry vycházející ze vzorku populace, nelze brát jako dogma. A jelikož v mém případě vzorek dat nebyl příliš obsáhlý a ještě byl specifický svou vysokou odborností zúčastněných, nelze mnou vyvozené závěry brát jako obecně platné za každé situace.

Podle mých závěrů se tedy může říct, že vyplnění jednoduchého formuláře, včetně pokročilých prvků, nedělá drtivé většině uživatelů žádné problémy. Pokud bych měl zmínit některé vlastnosti, které uživatelé často využívají, pak je to určitě procházení mezi poli pomocí klávesy `TAB`, a kopírování a vkládání textů pomocí klávesových zkratek. Kontextové menu po stisku pravého tlačítka vesměs nikdo nevyužívá, nejspíše proto, že uživatelé nejsou zvyklí na to, že webová aplikace může mít speciálně upravené kontextové menu stejně jako desktopová. Takže pokud má být v kontextovém menu nějaká speciální funkcionalita, měl by být o její existenci uživatel upozorněn.

Zajímavé je, že skrolování pomocí kolečka myši využívá pouze část uživatelů, a proto by vždy měla být možnost skrolovat i pomocí skrolovací lišty a případně jiných prvků anebo kláves.

Pokud máme pole, kde očekáváme, že uživatelé budou zadávat malá čísla, je vhodné ho opatřit prvkem `NumericUpDown`. Ten je uživateli poměrně využíván a v praxi se často, například v jiné formě, používá v e-shopech při zadávání počtu kusů daného výrobku. Samozřejmě by měla být zachována možnost zadat číslo přímo.

Při zobrazování okna bychom si měli uvědomit, že uživatelé v drtivé většině pro zavření okna automaticky zvolí tlačítko `OK` a neočekávají tedy vykonání nějaké jiné funkce. Proto, pokud chceme zobrazit okno vyžadující uživatelovo potvrzení, a nejedná se čistě jen o upozornění, není vhodné používat pro potvrzení dané netriviální události popis tlačítka `OK`.

Pokud jde o Tooltip, malou ikonu zobrazenou vedle pole, u které se zobrazí popisek po najetí myší, není tato uživateli příliš využívána. Proto, pokud má být dáno uživateli najevo nějaké důležitější sdělení, není vhodné pro jeho zobrazení použít Tooltip. Ten by měl sloužit čistě jen pro zobrazování upřesňujících informací. Při použití rozbalovacího seznamu bychom měli pamatovat na to, aby umožňoval jak přímé vybrání, tak vybrání pomocí zadání prvních několika písmen hledaného prvku.

Pokud jde o závislosti u formuláře, byla mezi uraženou vzdáleností a časem potřebným pro vykonání testu prokázána jen slabá závislost. Mezi počtem stisknutých kláves a časem potřebným pro vykonání testu nebyla prokázána žádná závislost.

Po vyhodnocení všech testů, které jsem provedl na rozhraních klasické menu, postraní panel a Ribbon jsem si v praxi ověřil, že platí pravidlo, kdy uživatelé u systému, který neznají, dávají ve většině případů přednost hlavní nabídce před panelem nástrojů s ikonami. Výjimku však tvořily ikony, které uživatelé dostatečně dobře znali a to například `Uložit` anebo `Tisk`. To platilo i pro rozhraní, kde je použito klasické menu a postranní panel. Ten je používán o poznání méně než hlavní menu. Pokud tedy uživatelé hledají příkaz, o kterém neví, kde je, dávají přednost

hlavní nabídce. Ribbon naopak funguje na jiném principu, ale nabízí například panel zvaný Quick Acces, obsahující jen pár nejzákladnějších funkcí. Tento byl uživateli hojně využíván.

Pokud bych měl tedy říct, který ze způsobů ovládání RIA aplikací je nejlepší, nedokázal bych na to objektivně odpovědět. Předně záleží na preferencích a zvyklostech uživatele. Při porovnání výsledků testů tedy nevyvstal jeden jediný vítěz a dá se říci, že si všechna rozhraní vedla poměrně vyrovnaně. I při porovnávání finálních časů potřebných k dokončení testovacích úkolů vyšlo najevo, že mezi jednotlivými rozhraními nejsou žádné výraznější rozdíly. Obecně jako nejrychlejší rozhraní se jevilo menu s postranním panelem, ale rozdíly proti ostatním nejsou natolik významné.

Při porovnání Ribbonu, jako zástupce nového rozhraní, se staršími rozhraními se dá říct, že si vede vesměs vyrovnaně. Na rozdíl od ostatních hrají u něj velký vliv použité ikony, zda jsou správně zvoleny a jsou dostatečně názorné. Ribbon je zároveň obecně prezentován jako způsob ovládání aplikací přizpůsobený běžným uživatelům, i převážně méně zkušeným až středně zkušeným, kteří však v našem testu byli zastoupeni v nepoměrně menším počtu oproti zkušeným uživatelům. Zároveň je to rozhraní poměrně nové, a tak se s ním někteří testeři možná setkali poprvé.

Při zkoumání zavilostí mezi rozhraními byla objevena jen slabá závislost, a to ta, že s uraženou vzdáleností roste čas potřebný pro vykonání testu. Je to poměrně logické, neboť to značí, že čím více uživatelé myší hledají v menu, tím déle jim trvá splnění daného úkolu. Závislost mezi použitím klávesnice a rychlostí vykonání testů však nebyla prokázána vesměs žádná anebo velmi slabá.

Při vyhodnocení výsledků vlastností, které jsem testoval jak při testech s formuláři tak i s rozhraními, vyšlo, že např. při zadávání data část uživatelů preferuje přímé zadání a část pomocí kalendáře. Proto je vhodné, aby bylo vždy dostupné zadání oběma způsoby. Obdobné je to při procházení mezi poli, i tam část uživatelé dává přednost klávese `TAB` a část přímému výběru pole pomocí kliknutí myši.

Vyhodnocením posledního testu jsem získal představu, jak uživatelé pracují s aplikacemi založenými na bázi tabulek. Je vidět, že uživatelé jsou z programů jako např. Excel zvyklí na určité chování, a tak například ve velké míře používají klávesnici. Proto je vhodné, aby tato standardní chování podporovaly i RIA aplikace jim podobné. Jde hlavně o klávesové zkratky pro vyjmutí, vložení a kopírování, mazání a pro multi-select pomocí `Ctrl` i `Shift`. Stejně jako v případě formuláře, skrolování pomocí kolečka myši využívá pouze asi polovina uživatelů, a proto by vždy měla být možnost skrolovat i pomocí skrolovací lišty, případně jiných prvků anebo kláves.

Používání pokročilých funkcí, jako seřazení kliknutím na hlavičku sloupce anebo použití filtrů, bylo až u zhruba 90% uživatelů. Obě funkce byly používány přibližně stejně často, ale když měli uživatelé na výběr, dali přednost filtrům. Dále bylo ověřeno, že mezi počtem použitých filtrů nebo seřazení a času vykonání testu není žádná závislost. Uživatelům nedělalo problémy ani pokročilé použití filtrů, a to i když podmínky filtrů byly v angličtině. Opět ale musíme brát v potaz specifický vzorek testerů. Části uživatelů však dělalo problém použité filtry poté opět

odstranit. Měli bychom si připomenout, že ve všech testech bylo testováno rozhraní, které uživatelé viděli poprvé. Takže závěry se vždy vztahují pouze na první uživatelskou zkušenost s daným rozhraním. Při dlouhodobé a časté práci s daným rozhraním se tedy závěry mohou lišit.

8.1. Budoucnost RIA aplikací

Předpovídat budoucnost je vždy ošidné a v oblasti informačních technologií, s rapidním vývojem, to platí dvojnásob. Proto budoucnost RIA aplikací je jen hádáním na základě aktuálních informací. Rád bych však zkusil nastínit možnosti, kterými by se mohli RIA aplikace ubírat do budoucna.

Pokud jde tedy o RIA aplikace jako takové, jedním z jejich logických následovníků by mohly být RCA aplikace, neboli Rich Cloud Application (bohaté internetové aplikace v cloudu). Jde o aplikace, které se pro uživatele tváří jako klasické RIA aplikace, ale hlavní rozdíl u nich je v plně duplexní komunikaci se serverem. Tradiční vztah klient-server tedy může být nahrazen vztahy mezi uživateli s možností různých rolí (běžní uživatelé, administrátoři apod.). Na rozdíl od klasických cloudových aplikací bohatost je v tomto případě na klientské straně.

Pokud jde o technologie, tak se nejspíše bude čím dál tím více prosazovat HTML5. Hlavně na úkor AJAXu, ale i ostatních technologií. Na vývoj obsáhlých bussines RIA aplikací však stále budou vhodnější jiné technologie jako Flex a Silverlight. Technologie JavaFX se stále nijak výrazně neprosadila a musela by přijít s nějakou výraznou změnou, aby to v budoucnosti zvrátila.

Již v dnešní době začínají weby používat určité HTML5 prvky a i zájem o tuto technologii je u vývojářů poměrně vysoký. Stále je však problém v neexistující finální specifikaci, která má být podle různých předpovědí hotová až v roce 2022. [33]. Již nyní však nejnovější prohlížeče implementují různé prvky z HTML5, avšak každý jiné a občas i různě.

Pokud jde o trendy RIA aplikací, jedním z nich by určitě mohla být personalizace. Každý uživatel je jiný, a tak je dobré mu dát možnost zvolit si svůj vlastní styl zobrazení, ovládání atd., který mu vyhovuje. Dalším z trendů jsou jednoduché, ale sofistikované aplikace. Jde tedy o to nabídnout aplikace, které splní to, co uživatel od nich očekává, ale zároveň nebudou na uživatele klást velké nároky na obsluhu. Chybějící informace, které by musel uživatel zadat, nahradí sofistikovanost programu. Samozřejmě, ne vždy je možné toho dosáhnout. S budoucím rozvojem výkonu a přenosových kapacit sítí je také možno nabídnout uživateli opravdu bohatý vizuální zážitek, a to i pomocí 3D zobrazení. Jak klasické 3D zobrazení, které se v dnešní době už používá například u mapových aplikací pro zobrazení budov při velkém přiblížení, tak i stereoskopické 3D.

V dnešní době čím dál tím více uživatelů k prohlížení internetu používá mobilní zařízení. S klesajícími cenami zařízení a lepší dostupností mobilního internetu tento počet do budoucna určitě ještě poroste. Proto jednou z důležitých vlastností RIA aplikací určitě bude umožnit snadný přístup a používání i uživatelům mobilních platforem. To bude například velký problém pluginových technologií, které často pro podobná zařízení stále nemají potřebnou podporu.

9. Závěr

Cílem této práce bylo zmapování a ověření možností tvorby bohatých internetových aplikací v prostředí internetu. Práce byla rozdělena do tří fází. V první jsem porovnal a zhodnotil dostupné technologie pro tvorbu bohatých internetových aplikací. Technologie byly hodnoceny z několika pohledů, a to i s přihlédnutím na požadavky kladené na RIA aplikace do budoucna. V druhé fázi jsem vyhodnotil nejčastější prvky a rozhraní vyskytující se v bohatých internetových aplikacích a vypracoval jsem vhodné úkoly pro jejich otestování. Úkoly měly prvky a rozhraní otestovat jak z pohledu uživatelské přívětivosti, tak snadnosti na ovládání. Následně jsem podle nich navrhl a naimplementoval pomocí technologie Silverlight testovací aplikaci, která sloužila pro vzdálené uživatelské testování. Ve třetí fázi byla data, získaná pomocí uživatelského testování, statisticky zpracována a vyhodnocena. V závěru jsem zhodnotil nejlepších postupy pro tvorbu bohatých internetových aplikací s ohledem na výsledky analýz a zároveň i budoucích trendů.

S výsledky mé práce byla také seznámena společnost NetDirect, která je shledala zajímavými a má představu o využití výsledků i testovacích konceptů vůči svým produktům a zákazníkům. Společnost NetDirect je dodavatelem e-business aplikací s významným tržním podílem na českém a slovenském trhu a úzkou specializací na technologie firmy Microsoft, které jsem i já využil při implementaci testovací aplikace.

10. Použitá literatura

1. Macromedia Flash MX—A next-generation rich. [Online] 2002. http://download.macromedia.com/pub/flash/whitepapers/richclient.pdf.

2. Technologie a RIA. [Online] 2011. http://silverlight.cs.vsb.cz/01-technologie-a-ria.aspx.

3. **Pichlík, Roman.** Rich Internet Application. *Interval.cz.* [Online] 2005. http://interval.cz/clanky/rich-internet-application/.

4. World Wide Consortium. [Online] 2011. http://www.w3.org/.

5. HTML5. *Wikipedia.* [Online] 2011. http://en.wikipedia.org/wiki/HTML5_video#Table.

6. **Løvlie, Anders Sundnes.** browser support for html5 audio. *The Textopia blog.* [Online] 2010. http://textopiablog.wordpress.com/2010/06/25/browser-support-for-html5-audio/.

7. Rich Internet Application Market Share. *StatOwl.* [Online] 2010. http://www.statowl.com/custom_ria_market_penetration.php.

8. Microsoft Releases Silverlight 2 On Tuesday, Oct. 14. *liveside.net.* [Online] 2008. http://www.liveside.net/2008/10/13/microsoft-releases-silverlight-2-on-tuesday-oct-14/.

9. Get Microsoft Silverlight. [Online] 2011. http://www.microsoft.com/getsilverlight/Get-Started/Install/Default.aspx.

10. Eclipse Tools for Microsoft Silverlight. [Online] 2011. http://eclipse4sl.org/.

11. Flash Player penetration. *Adobe Software.* [Online] 2011. http://www.adobe.com/products/player_census/flashplayer/.

12. Rich Internet Application Statistics. [Online] 2011. http://riastats.com.

13. System requirements. *Adobe Software.* [Online] 2011. http://www.adobe.com/products/flashplayer/systemreqs/.

14. Release Notes Moonlight4 Preview. *Mono project.* [Online] 2011. http://www.mono-project.com/Release_Notes_Moonlight4_Preview.

15. Software and System Requirements for JavaFX 1.3 Technology. [Online] 2011. http://download.oracle.com/javafx/1.3/reference/system-requirements/system-requirements-1-3.html#nb65ide.

16. Top 5 Operating Systems. *StatCounter GlobalStats.* [Online] 2010. http://gs.statcounter.com/#os-ww-monthly-201002-201102-bar.

17. Operating Systems Market Share. *StatOwl.* [Online] 2010. http://www.statowl.com/operating_system_market_share.php.

18. **ERNS, Timo.** Performance Analysis and Acceleration for Rich Internet Application Technologies. *Ulm, 2010. 137 s. Diplomová práce. University of Ulm.* [Online] 2010. http://www.timo-ernst.net/misc/riabench-start/results/usecase/win-maintest-results.pdf.

19. About Flex Builder perspectives . *Adobe Software.* [Online] 2011. http://livedocs.adobe.com/flex/3/html/help.html?content=intro_workbench_4.html.

20. Silverlight Overview. *Silverlight.* [Online] 2011. http://www.silverlight.net/getstarted/overview.aspx.

21. **Danijel Stulic.** Multi-threading options in Rich Internet Applications . *Silverlighter.* [Online] 2009. http://stulic.blogspot.com/2009/09/multi-threading-options-in-rich.html.

22. JavaFX 2.0 Roadmap. *JavaFX.* [Online] 2011. http://javafx.com/roadmap/.

23. Mobile vs. Desktop in North America. *StatCounter GlobalStats.* [Online] 2010. http://gs.statcounter.com/#mobile_vs_desktop-na-monthly-201002-201102.

24. **Jobs, Steve.** Thoughts on Flash. *Apple Inc.* [Online] 2010. http://www.apple.com/hotnews/thoughts-on-flash/.

25. Adobe Flash Player 10.1 Coming To Windows Phone 7, BlackBerry, and Others. *The Tech Journal.* [Online] 2010. http://thetechjournal.com/electronics/computer/software/adobe-flash-player-10-1-coming-to-windows-phone-7-blackberry-and-others.xhtml.

26. Certified devices. *Adobe Software.* [Online] 2010. http://www.adobe.com/flashplatform/certified_devices/.

27. IE9 coming to Windows Phone 7 much sooner than expected, support Silverlight? *WMPoweruser.com.* [Online] 2011. http://wmpoweruser.com/ie9-coming-to-windows-phone-7-much-sooner-than-expected-support-silverlight/.

28. **A. Cooper, R. Reimann, D. Cronin.** *About Face 3: The Essentials of Interaction Desing.* Indiana : Wiley Publishing, Inc., 2007. 978-0-470-08411-3.

29. **Cipan, Vibor.** *Silverlight 4 User Interface Cookbook.* Olton : Packt Publishing, 2010. 978-1847198860.

30. RadControls for Silverlight. *Telerik.* [Online] 2010. http://www.telerik.com/products/silverlight.aspx.

31. **Troelsen, Andrew.** *Pro Expression Blend 4.* New York : Apress, 2011. str. 400. 978-1-4302-3377-0.

32. **Šimonová, Lenka.** Průvodce k programu Statgraphics, část 2. [Online] 2006. http://is457.vsb.cz/simonova/vyuka/statistikaI/Pruvodce_2.pdf.

33. **Šímová, Lenka.** Průvodce k programu Statgraphics, část 1. [Online] 2006. http://is457.vsb.cz/simonova/vyuka/statistikaI/Pruvodce_1.pdf.

34. HTML 5 Won't Be Ready Until 2022. Yes, 2022. *webmonkey.* [Online] 2008. http://www.webmonkey.com/2008/09/html_5_won_t_be_ready_until_2022dot_yes__2022dot/.